Administração de Sistemas Linux: **Redes e Segurança**

Pedro R. Torres Júnior
Christian Lyra Gomes
Francisco Marcelo M. Lima
Luiz Fernando Ramos Costa

A RNP – Rede Nacional de Ensino e Pesquisa – é qualificada como uma Organização Social (OS), sendo ligada ao Ministério da Ciência, Tecnologia e Inovação (MCTI) e responsável pelo Programa Interministerial RNP, que conta com a participação dos ministérios da Educação (MEC), da Saúde (MS) e da Cultura (MinC). Pioneira no acesso à Internet no Brasil, a RNP planeja e mantém a rede Ipê, a rede óptica nacional acadêmica de alto desempenho. Com Pontos de Presença nas 27 unidades da federação, a rede tem mais de 800 instituições conectadas. São aproximadamente 3,5 milhões de usuários usufruindo de uma infraestrutura de redes avançadas para comunicação, computação e experimentação, que contribui para a integração entre o sistema de Ciência e Tecnologia, Educação Superior, Saúde e Cultura.

RNP

Ministério da **Cultura**

Ministério da **Saúde**

Ministério da **Educação**

Ministério da **Ciência, Tecnologia e Inovação**

Administração de Sistemas Linux: **Redes e Segurança**

Pedro R. Torres Júnior
Christian Lyra Gomes
Francisco Marcelo M. Lima
Luiz Fernando Ramos Costa

Administração de Sistemas Linux: **Redes e Segurança**

Pedro R. Torres Júnior
Christian Lyra Gomes
Francisco Marcelo M. Lima
Luiz Fernando Ramos Costa

Rio de Janeiro
Escola Superior de Redes
2013

Copyright © 2013 – Rede Nacional de Ensino e Pesquisa – RNP
Rua Lauro Müller, 116 sala 1103
22290-906 Rio de Janeiro, RJ

Diretor Geral
Nelson Simões

Diretor de Serviços e Soluções
José Luiz Ribeiro Filho

Escola Superior de Redes

Coordenação
Luiz Coelho

Edição
Pedro Sangirardi

Revisão
Lincoln da Mata

Coordenação Acadêmica de Administração de Sistemas
Sergio Ricardo Alves de Souza

Equipe ESR (em ordem alfabética)
Celia Maciel, Cristiane Oliveira, Derlinéa Miranda, Edson Kowask, Elimária Barbosa, Lourdes Soncin, Luciana Batista, Luiz Carlos Lobato, Renato Duarte e Yve Abel Marcial.

Capa, projeto visual e diagramação
Tecnodesign

Versão
1.2.0

Este material didático foi elaborado com fins educacionais. Solicitamos que qualquer erro encontrado ou dúvida com relação ao material ou seu uso seja enviado para a equipe de elaboração de conteúdo da Escola Superior de Redes, no e-mail info@esr.rnp.br. A Rede Nacional de Ensino e Pesquisa e os autores não assumem qualquer responsabilidade por eventuais danos ou perdas, a pessoas ou bens, originados do uso deste material.
As marcas registradas mencionadas neste material pertencem aos respectivos titulares.

Distribuição
Escola Superior de Redes
Rua Lauro Müller, 116 – sala 1103
22290-906 Rio de Janeiro, RJ
http://esr.rnp.br
info@esr.rnp.br

Dados Internacionais de Catalogação na Publicação (CIP)

A238 Administração de sistemas Linux: redes e segurança / Pedro R. Torres Júnior ... [et. al.]. – 1. ed. rev. – Rio de Janeiro: RNP/ESR, 2012.
228 p. : il. ; 28 cm.

Bibliografia: p. 225-227.
ISBN 978-85-63630-21-6

1. Linux (Sistema operacional de computador). 2. Serviços de diretório (tecnologia de redes de computador). 3. Redes de computadores – Medidas de segurança. I. Torres Júnior, Pedro R. II. Título.

CDD 005.8

Sumário

Escola Superior de Redes

A metodologia da ESR **xiii**

Sobre o curso **xiv**

A quem se destina **xv**

Convenções utilizadas neste livro **xv**

Permissões de uso **xv**

Sobre os autores **xvi**

1. Introdução à administração de redes e arquitetura TCP/IP (parte 1)

Objetivos e funções da administração de redes **1**

Arquitetura TCP/IP **2**

Modelo ISO/OSI **3**

- Tipos de camada **4**

Modelo TCP/IP **6**

- Encapsulamento **7**
- Nível de enlace: Ethernet **7**
- Nível de rede: protocolo IP **9**

Datagrama IP **9**

- MTU e Fragmentação **10**

Endereçamento IP **11**

- Classes de endereços **11**
- Máscaras de rede **12**
- Endereços especiais **13**
- Sub-redes **14**
- CIDR **17**

Resolução de endereços 19

Atribuição de endereços (IP) 19

Roteiro de Atividades 1 21

Atividade 1.1 – Entendendo o modelo ISO/OSI 21

Atividade 1.2 – Visualizando a configuração de rede em uma máquina Linux 21

Atividade 1.3 – Entendendo a tabela ARP 21

Atividade 1.4 – Cabeçalho Ethernet 21

Atividade 1.5 – Fragmentação 21

Atividade 1.6 – Criando endereços de rede ou sub-rede 22

Atividade 1.7 – Notação CIDR 22

Atividade 1.8 – Faixa de endereços válidos na internet 22

Atividade 1.9 – Entendendo a máscara de rede 22

Atividade 1.10 – Endereçamento dinâmico 23

Atividade 1.11 – Resolução de nomes 23

2. Introdução à administração de redes e arquitetura TCP/IP (parte 2)

Roteamento 25

- Tabela de roteamento 26
- Exemplo de topologia 26
- Rotas estáticas x rotas dinâmicas 27
- Tempo de vida (TTL) 27

Algoritmo de roteamento 28

- Internet Control Message Protocol (ICMP) 28

Ping 31

Traceroute 31

Nível de transporte 32

- Portas 32

Nível de transporte: protocolo UDP 33

Nível de transporte: protocolo TCP 33

- Início e término de uma conexão 34

Nível de transporte: conexão virtual 35

Nível de transporte 36

Arquitetura cliente/servidor 36

Roteiro de Atividades 2 **39**

Atividade 2.1 – Roteamento **39**

Atividade 2.2 – Problema na tabela de roteamento **39**

Atividade 2.3 – Problema na tabela de roteamento **39**

Atividade 2.4 – Protocolo ICMP **40**

Atividade 2.5 – Comandos ping e traceroute **40**

Atividade 2.6 – Cabeçalho TCP **40**

Atividade 2.7 – Daemons **40**

Atividade 2.8 – Testando a conexão **40**

Atividade 2.9 – Conexões virtuais **40**

Atividade 2.10 – Testando serviços **40**

3. Configurando uma rede TCP/IP

Equipamentos de rede **41**

Hub **41**

Switch **41**

Configurando endereços e rotas **43**

Configurando endereços IP **43**

Configurando uma interface de rede **43**

Exemplos do comando *ifconfig* **44**

Configurando uma interface de rede **47**

Configurando endereços e redes **49**

Formas de conexão à internet **51**

Resolução de nomes **52**

Arquivo /etc/nsswitch.conf **52**

Arquivo /etc/hosts **53**

Arquivo /etc/resolv.conf **53**

Ferramentas para resolução de nomes **54**

DNSSEC **55**

Atribuição dinâmica de endereços IP **57**

ISC DHCP Client **58**

IPv6 **61**

Roteiro de Atividades 3 **63**

Atividade 3.1 – Diferença entre switches e hubs **63**

Atividade 3.2 – Comando *ifconfig* **63**

Atividade 3.3 – Comando *ip* **63**

Atividade 3.4 – Comando *route* **63**

Atividade 3.5 – Comando *ip* **63**

Atividade 3.6 – Rota inválida **63**

Atividade 3.7 – Configuração de rotas **64**

Atividade 3.8 – Resolução de nomes **64**

Atividade 3.9 – DNSSEC **64**

Atividade 3.10 – Atribuição dinâmica de endereços **64**

4. Verificando a configuração da rede

Tabela de roteamento **65**

Comando traceroute **67**

Comando ping **68**

Comando mtr **69**

Verificação do estado do link **69**

Comando arp **70**

Verificação da configuração das interfaces **70**

Sniffers **72**

BPF **72**

libpcap **72**

tcpdump **72**

Ethereal (Wireshark) **73**

Verificação de serviços **73**

Sockets **74**

Comando lsof **77**

Comando nmap **78**

Roteiro de Atividades 4 81

Atividade 4.1 – Tabela de Roteamento **81**

Atividade 4.2 – Comando *ifconfig* e/ou *ip link* **81**

Atividade 4.3 – Comando tcpdump **81**

Atividade 4.4 – Wireshark (Ethereal) **82**

Atividade 4.5 – Comandos *netstat* e/ou *lsof* **82**

Atividade 4.6 – Serviços **82**

Atividade 4.7 – Comando nmap **82**

Atividade 4.8 – Identificando serviços com o nmap **82**

Atividade 4.9 – Monitoramento de tráfego de rede **83**

5. Segurança – Introdução

Segurança **85**

Conexões de entrada: gerência e controle de acesso **86**

Iniciar e parar serviços **87**

Daemon inetd **88**

tcpwrapper **89**

tcpdmatch e tcpdchk **91**

Daemon xinetd **91**

Secure shell (SSH) **93**

Configurando o daemon sshd **94**

Chaves públicas **95**

Cliente SSH **96**

Uso de chaves públicas com o SSH **96**

Logs **98**

Análise de arquivos de log **98**

Logs remotos **99**

Analisadores de logs **100**

Logcheck **100**

RootKit Hunter **102**

Procedimentos de detecção de invasão **103**

Roteiro de Atividades 5 105

Atividade 5.1 – Run Level **105**

Atividade 5.2 – Run Level (continuação) **105**

Atividade 5.3 – XInetd **105**

Atividade 5.4 – tcpwrapper **105**

Atividade 5.5 – sshd **105**

Atividade 5.6 – sshd: chave RSA **105**

Atividade 5.7 – SCP **106**

Atividade 5.8 – Logs **106**

Atividade 5.9 – syslogd **106**

Atividade 5.10 – Procurando por indícios de invasão **106**

6. Segurança – Sistema Operacional

Procedimentos para instalação **107**

Particionando o sistema **109**

Política de segurança **111**

Fortalecendo o sistema **112**

Atualizações de segurança **112**

Configurar senha do Grub **113**

Restringindo o uso do console **113**

Desativando a reinicialização do sistema **114**

Implementando quotas de disco **115**

Fornecendo acesso seguro ao usuário **117**

Ativar o suporte MD5 nas aplicações de login e SSH **117**

Criando o grupo wheel **118**

Limitando o uso de recursos: o arquivo /etc/security/limits.conf **119**

Ações de login do usuário **119**

Usando o sudo **120**

Desativação de acesso administrativo remoto **120**

Configurando umask **120**

Limitando o conteúdo que os usuários podem ver e acessar **121**

Limitando o acesso a outras informações de usuários **121**

Restringindo o acesso de usuários **122**

Logout de usuários ociosos **122**

Segurança do kernel **122**

Proteção da Libsafe **122**

Removendo funcionalidades **122**

Sistema de arquivos proc **123**

Fazendo uma cópia de segurança do sistema **124**

chroot **124**

Instalação e configuração **124**

Iniciando e testando o funcionamento **125**

Instalação e configuração **125**

Iniciando e testando o serviço **126**

Tripwire **127**

Integridade do sistema de arquivos com Tripwire **127**

Instalação e configuração **127**

Editando e gerando o arquivo de configuração **128**

Editando e gerando o arquivo de políticas **128**
Iniciando e testando o serviço **128**

Roteiro de Atividades 6 **129**

Atividade 6.1 – Elaborando uma política de segurança **129**

Atividade 6.2 – Antes e durante a instalação **129**

Atividade 6.3 – Fortalecendo a segurança **129**

Atividade 6.4 – Fornecendo acesso seguro ao usuário **130**

Atividade 6.5 – Utilizando sudo **130**

Atividade 6.6 – Calculando e testando umask **130**

Atividade 6.7 – Limitando o acesso a outras informações de usuários **130**

Atividade 6.8 – Restringindo o acesso de usuários **130**

Atividade 6.9 – Implementando o logout automático de usuários ociosos **130**

Atividade 6.10 – Utilizando o chroot **131**

Atividade 6.11 – Utilizando o portsentry **131**

Atividade 6.12 – Utilizando o tripwire **131**

7. Segurança – Firewall

Firewall **133**
Tipos de firewalls **134**

Netfilter **135**
Regras **135**
Estados **135**

Chains **136**
Tabelas **137**

Configuração do firewall **137**
Políticas de firewall **137**

Módulos do kernel **138**

Manipulação de regras **139**

Padrões de casamento **140**

Listagem de regras **142**

Manipulação de chains **143**
Ajuste da política padrão de uma chain **143**
Criar e remover chains **143**
Utilizar chains de usuário **144**
Limpar chains **144**

Habilitando o repasse de pacotes **145**

Ajuste do número de conexões registradas **145**

Dicas **146**

Roteiro de Atividades 7 147

Atividade 7.1 – Módulo ip_conntrack **147**

Atividade 7.2 – Política padrão **147**

Atividade 7.3 – Firewall Stateful **147**

Atividade 7.4 – Firewall de host **147**

Atividade 7.5 – Liberando serviços no firewall **148**

Atividade 7.6 – Criando chain **148**

Atividade 7.7 – Firewall de rede **148**

Atividade 7.8 – Firewall de rede (continuação) **148**

Atividade 7.9 – Worm **149**

Atividade 7.10 – DMZ **149**

8. Interconexão de redes

Tradução de endereços de redes **151**

Tipos de NAT **151**

Source NAT (SNAT) **152**

Configuração do SNAT **153**

Configuração do DNAT **154**

Roteamento avançado **155**

Túneis **156**

Comandos de configurações **156**

Comando ip tunnel **156**

Túnel IP sobre IP (IPIP) **157**

Uso do túnel IPIP **158**

Túnel GRE **159**

Túnel SIT **160**

Tabelas de roteamento **161**

Entradas na tabela de roteamento **162**

Routing Policy Database (RPDB) **164**

Tipos de regras **165**

Roteiro de Atividades 8 **167**

Atividade 8.1 – SNAT **167**

Atividade 8.2 – SNAT (continuação) **167**

Atividade 8.3 – DNAT **167**

Atividade 8.4 – DNAT (continuação) **167**

Atividade 8.5 – Redirect **168**

Atividade 8.6 – Túnel IPIP **168**

Atividade 8.7 – MTU **168**

Atividade 8.8 – Tabela de Roteamento **168**

Atividade 8.9 – Testando rotas **169**

Atividade 8.10 – Roteamento pela origem **169**

9. VPN e protocolos de tunelamento de nível 2

Virtual Private Network (VPN) **171**

Proteção da VPN **172**

Arquitetura VPN de acesso remoto **172**

Arquitetura VPN intranet e extranet **173**

Protocolos de tunelamento – PPP/SLIP **174**

Protocolo PPP **174**

Pacotes LCP **176**

Protocolo SLIP **177**

Protocolo GRE **178**

Protocolo PPTP **179**

Protocolo L2F **180**

Protocolo L2TP **181**

Tunelamento L2TP **181**

Protocolo IPSec **182**

VLAN **182**

Qualidade de Serviço (QoS) – MPLS **185**

Roteamento **186**

Encaminhamento dos pacotes **187**

Comutação **188**

Componente de controle **188**

Componentes de encaminhamento **188**

Label Switching **188**

Informação de distribuição das etiquetas **190**

Roteador de Borda **191**

Roteiro de Atividades 9 195

Atividade 9.1 – Instalação do OpenVPN **195**

Atividade 9.2 – Utilizando certificados digitais **198**

10.IPSec

Introdução ao IPSec **207**

Associações de segurança **208**

Modos de operação **209**

Protocolo AH (IP Authentication Header) **210**

Formato do AH (Authentication Header) **211**

Protocolo ESP (IP Encapsulating Security Payload) **212**

Formato do ESP **212**

Modos de funcionamento do ESP **213**

Gerenciamento de chaves **213**

Gerência automática de chaves **214**

Principais conceitos sobre o IKE **215**

Algoritmo Diffie-Hellman (DH) **216**

Visão geral do funcionamento do protocolo **217**

Network Address Translation (NAT) Transverso **219**

Roteiro de Atividades 10 221

Atividade 10.1 – Criando o firewall **221**

Atividade 10.2 – Configurando as interfaces de rede **222**

Atividade 10.3 – Configurando IPSec no IPCop **224**

Atividade 10.4 – Configurando regras de firewall **224**

Bibliografia 225

Escola Superior de Redes

A Escola Superior de Redes (ESR) é a unidade da Rede Nacional de Ensino e Pesquisa (RNP) responsável pela disseminação do conhecimento em Tecnologias da Informação e Comunicação (TIC). A ESR nasce com a proposta de ser a formadora e disseminadora de competências em TIC para o corpo técnico-administrativo das universidades federais, escolas técnicas e unidades federais de pesquisa. Sua missão fundamental é realizar a capacitação técnica do corpo funcional das organizações usuárias da RNP, para o exercício de competências aplicáveis ao uso eficaz e eficiente das TIC.

A ESR oferece dezenas de cursos distribuídos nas áreas temáticas: Administração e Projeto de Redes, Administração de Sistemas, Segurança, Mídias de Suporte à Colaboração Digital e Governança de TI.

A ESR também participa de diversos projetos de interesse público, como a elaboração e execução de planos de capacitação para formação de multiplicadores para projetos educacionais como: formação no uso da conferência web para a Universidade Aberta do Brasil (UAB), formação do suporte técnico de laboratórios do Proinfo e criação de um conjunto de cartilhas sobre redes sem fio para o programa Um Computador por Aluno (UCA).

A metodologia da ESR

A filosofia pedagógica e a metodologia que orientam os cursos da ESR são baseadas na aprendizagem como construção do conhecimento por meio da resolução de problemas típicos da realidade do profissional em formação. Os resultados obtidos nos cursos de natureza teórico-prática são otimizados, pois o instrutor, auxiliado pelo material didático, atua não apenas como expositor de conceitos e informações, mas principalmente como orientador do aluno na execução de atividades contextualizadas nas situações do cotidiano profissional.

A aprendizagem é entendida como a resposta do aluno ao desafio de situações-problema semelhantes às encontradas na prática profissional, que são superadas por meio de análise, síntese, julgamento, pensamento crítico e construção de hipóteses para a resolução do problema, em abordagem orientada ao desenvolvimento de competências.

Dessa forma, o instrutor tem participação ativa e dialógica como orientador do aluno para as atividades em laboratório. Até mesmo a apresentação da teoria no início da sessão de aprendizagem não é considerada uma simples exposição de conceitos e informações. O instrutor busca incentivar a participação dos alunos continuamente.

As sessões de aprendizagem onde se dão a apresentação dos conteúdos e a realização das atividades práticas têm formato presencial e essencialmente prático, utilizando técnicas de estudo dirigido individual, trabalho em equipe e práticas orientadas para o contexto de atuação do futuro especialista que se pretende formar.

As sessões de aprendizagem desenvolvem-se em três etapas, com predominância de tempo para as atividades práticas, conforme descrição a seguir:

Primeira etapa: apresentação da teoria e esclarecimento de dúvidas (de 60 a 90 minutos).
O instrutor apresenta, de maneira sintética, os conceitos teóricos correspondentes ao tema da sessão de aprendizagem, com auxílio de slides em formato PowerPoint. O instrutor levanta questões sobre o conteúdo dos slides em vez de apenas apresentá-los, convidando a turma à reflexão e participação. Isso evita que as apresentações sejam monótonas e que o aluno se coloque em posição de passividade, o que reduziria a aprendizagem.

Segunda etapa: atividades práticas de aprendizagem (de 120 a 150 minutos).
Esta etapa é a essência dos cursos da ESR. A maioria das atividades dos cursos é assíncrona e realizada em duplas de alunos, que acompanham o ritmo do roteiro de atividades proposto no livro de apoio. Instrutor e monitor circulam entre as duplas para solucionar dúvidas e oferecer explicações complementares.

Terceira etapa: discussão das atividades realizadas (30 minutos).
O instrutor comenta cada atividade, apresentando uma das soluções possíveis para resolvê-la, devendo ater-se àquelas que geram maior dificuldade e polêmica. Os alunos são convidados a comentar as soluções encontradas e o instrutor retoma tópicos que tenham gerado dúvidas, estimulando a participação dos alunos. O instrutor sempre estimula os alunos a encontrarem soluções alternativas às sugeridas por ele e pelos colegas e, caso existam, a comentá-las.

Sobre o curso

O objetivo desse curso é fornecer ao aluno o conhecimento prático-teórico necessário ao profissional de informática, e através de atividades práticas específicas aos ambientes de segurança e redes, as ferramentas necessárias para construção e manutenção de redes de computadores em ambientes seguros. O curso apresentará as principais atividades em Administração de Redes, além de sugerir políticas de segurança a serem implementadas nas diversas instituições com o intuito de manter os ambientes computacionais funcionais e seguros. Serão apresentados os fundamentos da arquitetura TCP/IP, apresentando sua pilha de protocolos e serviços oferecidos. As atividades práticas do curso envolvem desde a configuração e monitoramento da rede até a implementação de firewall, NAT, roteamento e tunelamento, utilizando ferramentas como Wireshark, Iptables, IPSec, OpenVPN e rkhunter, entre outras.

A quem se destina

O público-alvo é composto por profissionais da área de informática com experiência em administração de sistemas Linux, que serão os responsáveis, ou farão parte de uma equipe, por instalar e manter uma rede de computadores com ênfase em rede TCP/IP. Podem também participar outros profissionais de TI, tais com gerentes, programadores e analistas de suporte de sistemas.

Convenções utilizadas neste livro

As seguintes convenções tipográficas são usadas neste livro:

Itálico
Indica nomes de arquivos e referências bibliográficas relacionadas ao longo do texto.

```
Largura constante
```

Indica comandos e suas opções, variáveis e atributos, conteúdo de arquivos e resultado da saída de comandos. Comandos que serão digitados pelo usuário são grifados em negrito e possuem o prefixo do ambiente em uso (no Linux é normalmente # ou $, enquanto no Windows é C:\).

Conteúdo de slide
Indica o conteúdo dos slides referentes ao curso apresentados em sala de aula.

Símbolo
Indica referência complementar disponível em site ou página na internet.

Símbolo
Indica um documento como referência complementar.

Símbolo
Indica um vídeo como referência complementar.

Símbolo
Indica um arquivo de áudio como referência complementar.

Símbolo
Indica um aviso ou precaução a ser considerada.

Símbolo
Indica questionamentos que estimulam a reflexão ou apresenta conteúdo de apoio ao entendimento do tema em questão.

Símbolo
Indica notas e informações complementares como dicas, sugestões de leitura adicional ou mesmo uma observação.

Permissões de uso

Todos os direitos reservados à RNP.

Agradecemos sempre citar esta fonte quando incluir parte deste livro em outra obra.
Exemplo de citação: TORRES, Pedro *et al. Administração de Sistemas Linux: Redes e Segurança*. Rio de Janeiro: Escola Superior de Redes, RNP, 2013.

Comentários e perguntas

Para enviar comentários e perguntas sobre esta publicação:
Escola Superior de Redes RNP
Endereço: Av. Lauro Müller 116 sala 1103 – Botafogo
Rio de Janeiro – RJ – 22290-906
E-mail: info@esr.rnp.br

Sobre os autores

Pedro R. Torres Jr é professor da Universidade Federal do Paraná e coordenador técnico do ponto de presença da RNP no estado do Paraná. Também atua como coordenador técnico do ponto de troca de tráfego (IXP) da região e da Rede Metropolitana de Curitiba. Possui mais de 14 anos de experiência em sistema operacional Linux e mais de 12 anos de experiência com roteamento IP.

Christian Lyra Gomes é formado em Ciências da Computação e tem mestrado em Redes de Computadores pela Universidade Federal do Paraná. Teve seu primeiro contato com Linux em 1997 e em 2000 trabalhou na Conectiva Linux. Em 2001, foi contratado pelo Ponto de Presença da RNP no Paraná atuando na área de gerência de redes e sistemas. Em 2006 assumiu a coordenação administrativa do PoP-PR, cargo que exerce até hoje.

Francisco Marcelo M. Lima é certificado Project Management Professional (PMP) e Modulo Certified Security Officer (MCSO), Mestre em Engenharia Elétrica pela Universidade de Brasília (2009), Mestre em Liderança pela Universidade de Santo Amaro (2007) e pós-graduado em Segurança de Redes de Computadores pela Universidade Católica de Brasília (2003). Atualmente exerce as funções de Coordenador dos Cursos de Redes de Computadores e Segurança da Informação do IESB, e Analista em TI do MPOG cedido para a Controladoria-Geral da União/PR. Atua também como instrutor/revisor dos cursos de segurança e redes na ESR e instrutor/revisor dos cursos de planejamento estratégico (PDTI) e gestão de contratos de TI (GCTI) na ENAP. Possui mais de 15 anos de experiência na área de Ciência da Computação, com ênfase em Segurança da Informação, Redes e Construção de Software, tendo exercido funções como: Coordenador Geral de TI do INCRA (DAS 4); Coordenador do Curso de Segurança da Informação da Faculdade Rogacionista; Coordenador do Curso de Processamento de Dados e Segurança da Informação da Faculdade AD1, Analista em Segurança da empresa Módulo Security Solutions.

Luiz Fernando Ramos Costa é Pós-graduando em Engenharia de Software – UNISUL/SC, formado em Redes de Computadores pela Estácio de Sá, certificado LPI Linux, Novell CLA, DCTS, analista de TI do IFSC e analisa de suporte na POWERSolutions. Mais de 13 anos de experiência em administração de redes, desenvolvimento de softwares e como orientador de cursos para a certificação Linux Professional Institute, atuando principalmente na gerência de redes em grandes corporações e com programação de sistemas web.

Sergio Ricardo Alves de Souza possui mais de 35 anos de experiência na área de Administração e Suporte de Sistemas. Trabalhou em empresas como: Burroughs (UNISYS), ARSA (Infraero), Cobra Computadores, LNCC e outras. Consultoria e treinamento em empresas como: Fiocruz, Jardim Botânico, Museu Goeldi, Cefet-MG, IBM Brasil. Participou do desenvolvimento e implantação de cursos profissionalizantes em Informática em Petrópolis - FAETC. Participou do projeto de criação do Instituto Superior de Tecnologia em Ciência da Computação de Petrópolis. Possui “Notório Saber em Informática” pelo LNCC.

1

Introdução à administração de redes e arquitetura TCP/IP (parte 1)

objetivos

Conhecer as principais atividades de administração de redes; entender o princípio de funcionamento da arquitetura TCP/IP e seus principais protocolos; conhecer os modelos de referência ISO/OSI e TCP/IP; identificar as camadas e conhecer suas funcionalidades; entender os mecanismos de endereçamento IP e o funcionamento da resolução de endereços e nomes.

conceitos

Administração de redes e arquitetura TCP/IP.

Objetivos e funções da administração de redes

Desempenhar qualquer atividade para manter o bom funcionamento da rede.

- Planejar.
- Implementar.
- Monitorar.
- Proteger.
- Escalar.
- Auditar.

Um administrador de redes deve ser capaz de planejar, implementar, monitorar, proteger, escalar e auditar a rede, otimizando-a sempre que possível para melhor atender seus usuários (locais ou remotos). Além disso, também deve saber gerenciar os ativos de rede e manter sua infraestrutura, analisar a evolução da utilização dos recursos computacionais para fornecer dados necessários para escalar o ambiente e garantir a autenticidade, integridade, disponibilidade e confidencialidade dos dados.

Frente a tanta inovação tecnológica e ambientes tão heterogêneos, o administrador de redes também deve manter-se atualizado em relação às integrações dos ativos de rede, falhas de segurança identificadas, atualizações para corrigir problemas com a segurança ou mau funcionamento, inovações de hardware e software.

Atualmente a maioria dos ambientes exige interconexão com redes públicas ou parceiras. Sendo assim, é dever do administrador de redes zelar pela sua rede e pelo trânsito de dados

entre as redes. Sempre que possível, interagir com o par que mantém a rede remota para que as melhores práticas sejam comuns às pontas, evitando assim que pontos de fragilidade remotos possam comprometer seu ambiente computacional.

Funções do dia a dia do administrador:

- Fazer instalações físicas e lógicas.
- Implementar, monitorar e auditar a rede.
- Atualizar serviços e corrigir bugs ou falhas de segurança.
- Analisar e atender a incidentes de segurança.
- Pesquisar por novidades tecnológicas tanto para melhor atender ao usuário quanto para facilitar o gerenciamento da rede.
- Documentar topologia, procedimentos, atendimentos etc.

Para cumprir com suas atribuições de maneira satisfatória, o administrador de sistemas deve sempre realizar atividades preventivas, tais como:

- **Instalações físicas/lógicas**: o planejamento de uma nova rede ou da expansão de uma rede já instalada deve ser feito nos níveis físico e lógico, envolvendo aspectos desde a seleção da tecnologia de cabeamento e dos equipamentos de interconexão até o projeto de endereçamento e roteamento;
- **Implementação, monitoramento e auditoria da rede**: o administrador deve sempre estar atento ao funcionamento da rede, analisando logs (registros de eventos que ocorrem no sistema), identificando possíveis sintomas de problemas e analisando se o desempenho da rede está normal. Caso não esteja, ele deve avaliar as possíveis correções para a situação encontrada. A auditoria pode ser necessária para identificar, coibir e aplicar as devidas sansões aos responsáveis por maus comportamentos de pessoas ou equipamentos dentro da sua rede, de acordo com as políticas e leis vigentes;
- **Atualização de programas**: um ponto importante dessa atividade é assegurar a continuidade e a segurança da operação dos sistemas, isto é, os serviços críticos da instituição devem continuar operando normalmente. Para tanto, os aplicativos instalados devem ser sempre atualizados. Isso envolve um bom planejamento da atualização, podendo resultar inclusive em atividades fora do horário do expediente normal;
- **Segurança**: embora o Linux seja considerado um sistema muito seguro, com a expansão da comunicação entre máquinas e dos acessos externos (remotos), as redes tornaram-se cada vez mais vulneráveis. Por isso, o administrador deve sempre ficar atento, verificar os logs (registros de eventos que ocorrem no sistema) de segurança e rastrear tentativas de invasão.

Com o desenvolvimento da internet, a disponibilidade tornou-se item de qualidade de um serviço, além dos já conhecidos: performance, segurança e confiabilidade. Como o próprio nome diz, a disponibilidade indica o tempo que um serviço fica acessível para uso. O grau de disponibilidade também pode ser medido pelo seu inverso, ou seja, pelo tempo que o sistema esteve indisponível (por exemplo, 90 minutos por semana).

Para se manter atualizado sobre questões de segurança, **bugs** e backdoors (método que permite obter acesso a um computador remoto sem passar pelos métodos normais de autenticação), uma boa dica é frequentar com regularidade listas de discussão sobre o tema, bem como sites de hackers (termo utilizado para designar um especialista em computação).

Bug
É um erro no funcionamento comum de um software (ou também de hardware), também chamado de "falha na lógica programacional de um programa de computador". Pode causar discrepâncias no objetivo ou ainda a impossibilidade de realização de uma ação de um programa de computador ou apenas travar o sistema.

Arquitetura TCP/IP

Modelos conceituais:

- Ajudam a entender o funcionamento da rede.
- Classificam protocolos.
- Baseiam-se no conceito de divisão de camadas.

Tipos de modelos:

- ISO/OSI.
- TCP/IP.

O que chamamos simplesmente de rede é, na verdade, um complexo sistema composto por hardware, softwares, Sistemas Operacionais e protocolos. Para ajudar a entender esse sistema e facilitar a criação e a implementação de protocolos ou softwares, criaram-se modelos conceituais baseados na ideia de camadas como forma de dividir as várias partes do sistema e de organizar a grande quantidade de protocolos envolvidos.

Dois modelos conceituais se popularizaram: o ISO/OSI e o TCP/IP.

Embora o modelo OSI seja considerado ideal, sua adoção em larga escala foi dificultada por alguns fatores, como, por exemplo, a diversidade de protocolos. Já o modelo TCP/IP, que se baseia no ISO/OSI, tornou-se o padrão de fato, principalmente por minimizar o conjunto de protocolos e ter sido ampla e gratuitamente disseminado em ambientes Unix.

Modelo ISO/OSI

O Open Systems Interconnection (OSI) segue o conceito de camadas.

- Cada camada recebe e envia requisições à camada imediatamente superior ou inferior e se comunica com a mesma camada em outro host.
- Uma camada não oferece serviços suportados por outra camada.

Conceito de camadas:

- Os serviços oferecidos são feitos através de uma camada bem definida.
- Cada camada possui uma forma de identificação dos dados que, por sua vez, encapsulam dados das camadas superiores.

O Open Systems Interconnection (OSI) foi desenvolvido pela International Organization for Standardization (ISO) no início dos anos 60, quando se iniciou a discussão sobre processamento em rede. O conceito de camadas desse modelo parte das seguintes premissas:

- Uma camada recebe e envia requisições para outra imediatamente superior ou inferior e se comunica com a mesma camada em outro host (qualquer computador conectado na internet);
- Nenhuma camada oferece os serviços suportados por outra camada;
- Cada camada oferece seus serviços através de uma interface bem definida;
- Cada camada possui uma forma de identificação (endereçamento) de seus próprios dados que, por sua vez, encapsulam dados das camadas superiores.

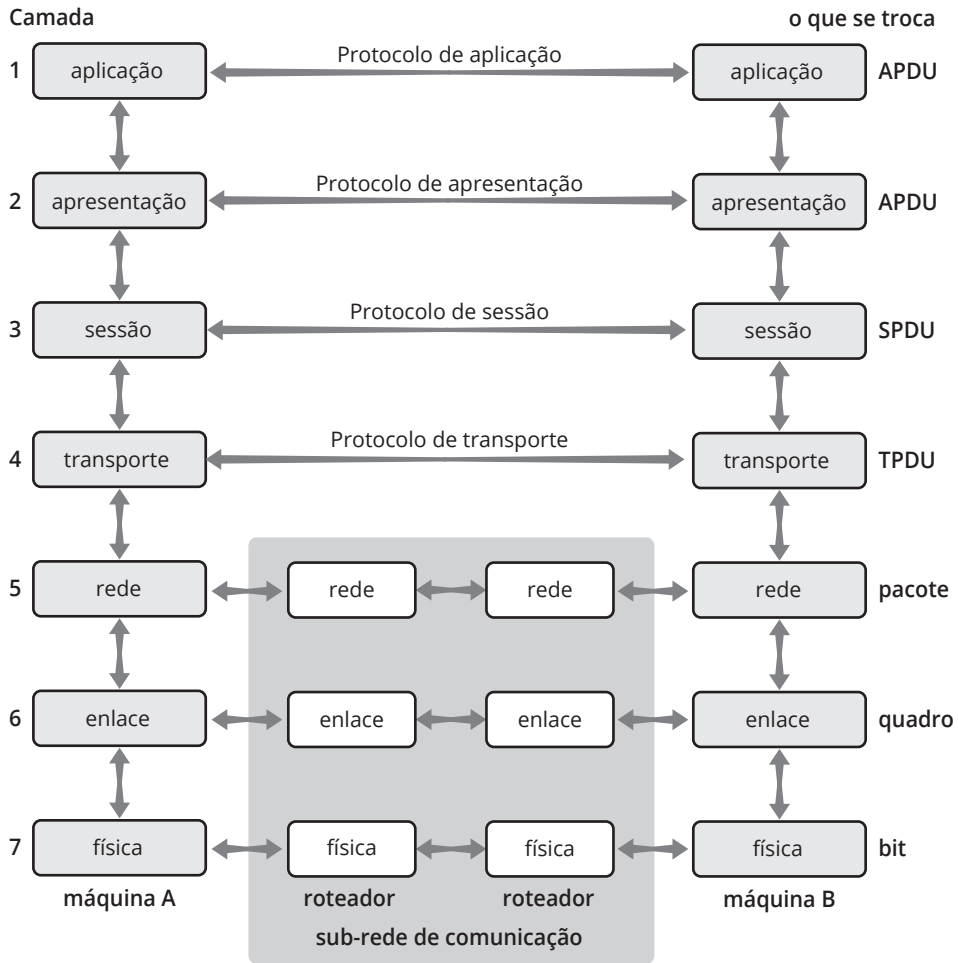


Figura 1.1
Em sete níveis, o modelo OSI e suas camadas.

O modelo OSI propõe uma estrutura de camadas (sete níveis) como referência para a arquitetura de protocolos de redes de computadores.

Tipos de camada

Física:

- Transforma bits (0 e 1) em sinais elétricos, ópticos e ondas de rádio.

Enlace:

- Agrupa os bits recebidos da camada física em quadros de dados.
  - Exemplo: quadro Ethernet.
- **Camada física (nível 1):** estabelece o canal de comunicação das máquinas com base no meio de transmissão (cabo coaxial, par trançado, fibra ótica etc.). Sua função é transformar bits (0 e 1) em sinais elétricos, ondas de rádio, micro-ondas etc. Para fins de aprendizado, pode-se considerar que essa camada é atualmente constituída pela placa de rede, que transforma o sinal lógico em elétrico, pelo cabo de rede e pelos conectores das máquinas;

- **Camada de enlace (nível 2)**: agrupa os bits recebidos do nível 1 em quadros de dados. Após se identificar como destino do quadro recebido e verificar que ele está livre de erros, a camada de enlace extrai suas informações de controle e envia os dados para a camada superior. Para isso, ela implementa mecanismos de identificação de início e fim dos quadros de dados, mecanismos de endereçamento, protocolos para acessar o meio e protocolos para detectar e opcionalmente corrigir erros de quadros.

Rede:

- Define e encaminha a unidade básica de transferência (datagrama).
  - Exemplo: IP.

Transporte:

- Pode assegurar que pacotes enviados serão recebidos.
- Pode implementar o controle de sequência e erros.
- Comunicação entre processos.

- **Camada de rede (nível 3):** responsável pela definição e pelo roteamento de datagramas (unidade básica de transferência de rede) entre máquinas que podem estar conectadas em redes físicas distintas. O roteamento é baseado em mecanismos de endereçamento globais que identificam cada máquina da rede de forma única. Uma vez que os pacotes trafegam através de redes com diferentes capacidades de transmissão, esse nível também pode implementar mecanismos de controle de congestionamento;
- **Camada de transporte (nível 4):** assegura que os pacotes enviados pela entidade do nível 4 da máquina-origem sejam recebidos pela entidade do nível 4 da máquina-destino. Para isso, pode implementar mecanismos de controle de sequência e de controle de erros que asseguram que os dados serão entregues na ordem correta, sem duplicações e erros. Também pode implementar um controle de fluxo para equilibrar as capacidades de processamento das máquinas-origem e destino. Esse nível fornece conectividade fim-a-fim, o que isola as camadas superiores dos aspectos de transmissão de dados.

Sessão:

- Provê mecanismos para lidar com funcionalidades necessárias à aplicação.
  - Exemplo: transferência de arquivos com checkpoints.

Apresentação:

- Trata o formato dos dados.

Aplicação:

- Dados gerados pela aplicação-origem e requisitados e/ou recebidos pela aplicação-destino.
  - Exemplo: e-mail.

- **Camada de sessão (nível 5):** fornece alguns mecanismos que lidam com a funcionalidade necessária para a aplicação, como controlar o diálogo entre os sistemas, na definição de uma disciplina de comunicação. Também pode implementar mecanismos de recuperação de falhas que permitem a retomada de transferências de dados do ponto onde foram interrompidas;
- **Camada de apresentação (nível 6):** realiza transformações nos dados relativas à compressão de textos, criptografia e conversão de formatos de representação. Se essa camada existisse no TCP/IP, a conversão de acentos e de números não precisaria ser executada através dos navegadores web;

- **Camada de aplicação (nível 7):** aqui os dados são essencialmente aqueles gerados pela aplicação-origem e requisitados e ou mesmo recebidos pela aplicação-destino. Pertencem a esse nível as aplicações de propósitos gerais, como transferência de arquivos, correio eletrônico e terminal remoto;

Lembre-se: o nível 1 apenas pegou um sinal físico e o passou para o nível 2 como um sinal lógico, interpretado de acordo com seus protocolos, sem se preocupar com o seu conteúdo.

Modelo TCP/IP

Aplicação:

- Similar às de aplicação, apresentação e sessão do ISO/OSI.

Transporte:

- Similar à de transporte do ISO/OSI.

Rede:

- Similar à de rede do ISO/OSI.

Enlace:

- Similar à de enlace do ISO/OSI.

Física:

- Similar à física do ISO/OSI.

O modelo TCP/IP surgiu com base em protocolos de algumas camadas já existentes e também como resultado prático de pesquisas em universidades. O TCP/IP implementa funcionalidades similares ao modelo OSI nas camadas de rede, transporte, sessão, apresentação e aplicação.

O modelo TCP/IP possui cinco camadas. É possível encontrar, na bibliografia, referências ao modelo TCP/IP com apenas quatro camadas (a camada física é suprimida):

- **Camada física (ou hardware)**: similar à do modelo ISO/OSI, transforma bits em sinais elétricos, ópticos etc., e os transmite através de um meio (um cabo, uma fibra ou mesmo pelo ar);
- **Camada de enlace (ou interface de rede)**: é a segunda mais baixa do modelo TCP/IP. Sua função também é similar à do modelo ISO/OSI, ou seja, é responsável por receber os quadros de dados de tipos de redes específicas (por exemplo, Ethernet, PPP e HDLC) e prepará-los para a camada superior (a camada de rede);
- **Camada de rede (ou de internet ou de inter-rede)**: responsável por transmitir pacotes de dados entre duas máquinas, possivelmente em redes diferentes. Seu principal protocolo é o Internet Protocol (IP). Nessa camada é feito todo o processo de roteamento de pacotes. Também é responsável pelas mensagens de controle e de erro (protocolo ICMP);
- **Camada de transporte**: responsável pela comunicação fim-a-fim de aplicativos. Enquanto na camada anterior temos a comunicação de máquina-a-máquina, nesta temos a comunicação de aplicativo-a-aplicativo. Os seus principais protocolos são o TCP e o UDP;
- **Camada de aplicação**: é a camada mais alta, onde são rodados os aplicativos com seus protocolos específicos, como transferência de arquivos, e-mail, vídeo etc.

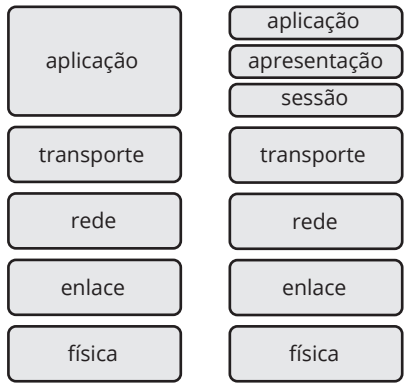


Figura 1.2
TCP/IP x ISO/OSI.

Encapsulamento

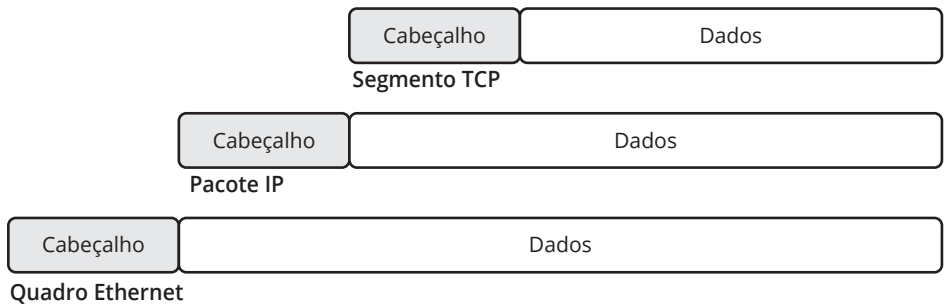


Figura 1.3
Funcionamento das camadas.

Antes de prosseguirmos no estudo das camadas de enlace e de rede, vamos estudar com mais detalhes o funcionamento e/ou relacionamento das camadas. Como visto anteriormente, cada camada possui uma forma de identificação e/ou endereçamento, e cada uma delas encapsula dados das camadas superiores. De uma forma geral, os dados que trafegam pelas camadas são organizados em pacotes (cada um contém um cabeçalho e uma área de dados). Na imagem anterior é mostrado um pequeno modelo com as camadas 2, 3 e 4.

Esse modelo permite que um pacote de uma determinada camada seja transportado por diferentes protocolos da camada inferior. Imagine um pacote IP com origem na máquina A, ligada em uma rede Ethernet, com destino à máquina B, situada em uma rede **Token Ring**. Essa transmissão se daria da seguinte forma:

Token Ring
Protocolo para redes locais. Um quadro chamado de Token circula por uma topologia em anel ou estrela para controlar o acesso ao meio compartilhado. Em inglês, "ring" quer dizer "anel" ou "círculo".

- A máquina A encapsularia o pacote IP em um quadro Ethernet e o transmitiria para um roteador com interfaces nas duas redes;
- Esse roteador extrairia o pacote IP do quadro Ethernet, o encapsularia em um quadro Token Ring e o transmitiria para a máquina B;
- A máquina B, por sua vez, extrairia o pacote IP do quadro Token Ring e o processaria.

Nível de enlace: Ethernet

- Padrão IEEE 802.3.
- Endereços de 48 bits.
- Controle de acesso ao meio e detecção de colisão (CSMA/CD).

O nível de enlace é o nível 2 do modelo ISO/OSI (que adotaremos como referência). A família de protocolos TCP/IP é capaz de funcionar sobre uma grande diversidade de tecnologias e protocolos de nível 2. Abordaremos aqui o padrão Ethernet, um dos mais populares para redes locais.

O padrão Ethernet (padrão IEEE 802.3) possui um sistema de endereçamento de 48 bits. Apesar de pertencer à camada de enlace, é conhecido como endereço físico ou ainda como endereço Medium Access Control (MAC – em inglês, Controle de Acesso ao Meio) e está associado ao dispositivo de interface de rede. Os fabricantes de dispositivos de rede compram blocos de endereços Ethernet e os designam em sequência aos seus produtos, o que em tese faria com que cada placa de rede possuísse um endereço único. No entanto, isso nem sempre acontece (veremos como contornar esse tipo de problema em outro Capítulo). O Sistema Operacional pode ler o endereço físico de cada dispositivo de rede (Ethernet). Esse endereço é representado separando-se os octetos por dois pontos, como no exemplo: 00:00:21:2C:E8:95.

Os endereços físicos podem ser de três tipos: unicast (de interface); multicast (identifica um grupo); ou broadcast (todos os computadores). Mais adiante veremos alguns protocolos que utilizam o broadcast para algumas funções especiais como, por exemplo, obter um endereço IP dinamicamente.

Formato do quadro

Um quadro Ethernet é composto de um preâmbulo, um cabeçalho, dados do usuário e um campo de Cyclic Redundancy Check (CRC) que serve para verificar a integridade do quadro.

Preâmbulo	Endereço de destino	Endereço de origem	Tamanho do quadro	Dados do usuário	CRC
8 octetos	6 octetos	6 octetos	2 octetos	46 - 1500 octetos	4 octetos

Figura 1.4
A estrutura de um quadro Ethernet em detalhes.

O preâmbulo serve apenas para sincronizar os relógios do receptor e do transmissor e não é considerado para efeito de cálculo do tamanho do quadro. Sendo assim, um quadro Ethernet pode ter tamanho mínimo de 64 octetos e máximo de 1.518 (incluindo o cabeçalho). Em algumas versões da tecnologia Ethernet, como na primeira desenvolvida pela Xerox, o campo "tamanho do quadro" é substituído por um campo chamado "type", de mesmo tamanho, mas com a função de identificar o protocolo de nível superior.

Cada tecnologia de rede utiliza um protocolo de acesso ao meio, muitas vezes entendido como uma subcamada do nível 2. Esses protocolos, como o Aloha, Carrier Sense Multiple Access (CSMA), CSMA/Collision Detection (CSMA/CD), além de outros, são utilizados para controle de acessos múltiplos. Utilizado na tecnologia Ethernet, o CSMA/CD possui um método para detectar colisão, ou seja, além de verificar se o canal está ocupado no instante da transmissão, também avisa se uma colisão ocorrer durante o tempo de acesso ao meio.

Quando duas máquinas na mesma rede local querem se comunicar, elas precisam primeiro descobrir o endereço físico do receptor. Para isso é utilizado o protocolo Address Resolution Protocol (ARP), que veremos mais adiante.

> A interface de rede só aceita quadros Ethernet endereçados para o seu próprio endereço físico ou para o endereço de broadcast. No entanto, a placa pode ser configurada para aceitar qualquer quadro Ethernet que passe no meio. Nesse caso, dizemos que a placa está em modo promíscuo.

Nível de rede: protocolo IP

Não confiável:

- O protocolo não implementa mecanismos de confirmação.
  - O emissor da mensagem não sabe se o pacote (ou datagrama) chegou ao destino

Não orientado à conexão:

- Não existe um mecanismo que indique se a ordem de recebimento dos pacotes foi a mesma de envio.
  - Os pacotes podem chegar fora de ordem.

Na camada de rede do modelo ISO/OSI, que corresponde à camada de internet do modelo TCP/IP, são implementados alguns protocolos, como IP, ICMP, ARP e RARP. Veremos detalhes do protocolo IP, como o formato do datagrama IP e as suas formas de endereçamento, na versão 4 do protocolo (IPv4).

Antes dos detalhes, há duas características importantes que devemos saber sobre o protocolo IP: ele não é confiável e não é orientado à conexão.

- **Não confiável**: o protocolo não implementa mecanismos de confirmação, ou seja, o emissor da mensagem não sabe se o pacote (ou datagrama) chegou ao destino;
- **Não orientado à conexão**: não existe um mecanismo que indique se a ordem de recebimento dos pacotes foi a mesma de envio, isto é, os pacotes podem chegar fora de ordem.

Datagrama IP

O datagrama IP possui um cabeçalho (header) IP que contém informações de controle do protocolo, como origem, destino e dados que está levando. A próxima figura ilustra o formato do cabeçalho:

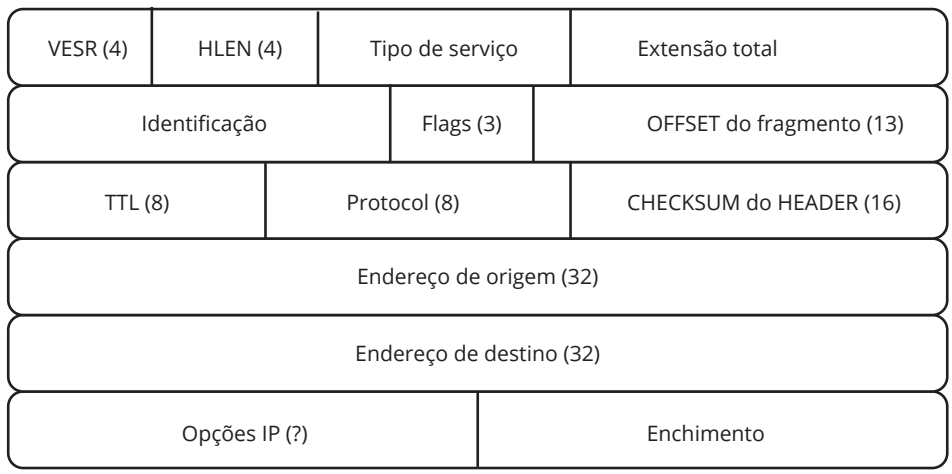


Figura 1.5 Informações de controle do protocolo no datagrama IP.

Descrição da utilização do cabeçalho IP:

- **Vers**: versão do protocolo IP (atualmente IPv4);
- **HLEN**: comprimento do cabeçalho em unidades de 32 bits;

- **Tipo de serviço**: indica a qualidade do serviço requerido. Por exemplo: prioridade, retardo, vazão e confiabilidade;
- **Comprimento total**: 2^16 - tamanho do pacote (64 K);
- **Identificação**: identificação dos pacotes (pacotes fragmentados têm a mesma identificação);
- **Flags**: DF – 1 bit indicando pedido para não fragmentar.
  MF – 1 bit indicando se existem ainda outros fragmentos (= 1) desse pacote se tratando do último fragmento (= 0);
- **Offset do fragmento**: posição do fragmento com relação ao datagrama original;
- **Tempo de Vida (TTL)**: evita que pacotes fiquem circulando indefinidamente na rede, sobrecarregando os roteadores. Cada roteador que processa o datagrama subtrai o valor do Tempo de Vida (TTL). Se o TTL atingir o valor 0 (zero), o datagrama é descartado e uma mensagem ICMP é enviada ao roteador que a originou, avisando que o tempo de vida do pacote expirou;
- **Protocol**: identifica o protocolo encapsulado no IP;
- **Checagem do cabeçalho**: mecanismo utilizado para detectar erros no cabeçalho;
- **Endereço de origem/destino**: endereços IP de origem e de destino;
- **Opções**: permitem teste e depuração da rede;
- **Enchimento**: usado para fazer um enchimento quando as opções não são múltiplas de 32 bits.

- Maximum Transmission Unit (MTU).
- Transmissão de um datagrama IP.
- O que é um quadro Ethernet?
  - Fragmentação.

MTU e Fragmentação

É importante destacar que um datagrama IP é encapsulado dentro de um quadro. Mas o que acontece quando a capacidade desse quadro é menor que o pacote IP que queremos enviar? Nesses casos, torna-se necessário fragmentar o pacote IP. É por isso que no cabeçalho existem campos que tratam de fragmentação. Por exemplo, se temos um Maximum Transmission Unit (MTU) – quadro que encapsula um datagrama IP – de 600 bytes e queremos enviar um pacote IP de 1400 bytes, esse pacote IP será fragmentado da seguinte forma:

- No primeiro quadro serão enviados 600 bytes, indicando-se no cabeçalho do IP que se trata do primeiro fragmento “setando” o campo “Offset” para 0 e a “Flag MF” para 1;
- No segundo quadro serão enviados mais 600 bytes, indicando-se no cabeçalho do IP que se trata do segundo fragmento “setando” o campo “Offset” para 600 e a “Flag MF” para 1;
- Por fim, no terceiro quadro serão enviados 200 bytes, indicando-se no cabeçalho IP que se trata do último fragmento “setando” o campo “Offset” para 1200 e a “Flag MF” para 0;

Considerações sobre fragmentação:

- A fragmentação permite a transmissão de pacotes IP de no máximo 64 kbytes sobre um quadro;
- Os datagramas fragmentados serão remontados apenas no destino final;

- Roteadores não fazem remontagem;
- Além do overhead para fragmentar, transmitir ou remontar, se um fragmento se perde, todo o datagrama é perdido.

Endereçamento IP

32 bits (representação em 4 decimais de 8 bits).

- 192.168.0.1 = 11000000.10101000.00000000.00000001

Protocolo ARP.

- Associação IP x Endereço físico.

O protocolo IP é responsável pelo endereçamento e roteamento dos pacotes. No TCP/IP, o endereço recebe o nome de endereço IP e é constituído por 32 bits que são representados na notação decimal com pontos, separados em 4 decimais de 8 bits. Por exemplo:

192.168.0.1 = 11000000.10101000.00000000.00000001

172.17.0.2 = 10101100.00010001.00000000.00000010

10.10.0.3 = 00001010.00001010.00000000.00000011

O endereço IP está associado ao nível de rede. O nível IP define uma associação entre um endereço IP e um endereço físico. Um determinado endereço IP pode ser associado somente a um único endereço físico. Essa associação será descrita mais detalhadamente quando abordarmos o protocolo ARP.

Cada endereço pode ser dividido em duas partes:

- NetId:
  - Conjunto dos bits que representa o endereço da rede.
- HostId:
  - Conjunto dos bits que representa a máquina dentro da rede.

Classes de endereços

Todas as máquinas de uma determinada rede compartilham um prefixo nos seus endereços IP. Esse prefixo é denominado endereço de rede e identifica apenas uma determinada rede conectada à internet. Implicitamente, o endereço de rede define o bloco de endereços IP que podem ser alocados para máquinas daquela rede. Cada endereço IP possui, logicamente, duas partes: NetId e HostId. No exemplo a seguir os bits do NetId são representados com a letra “n” e os bits do HostId, com a letra “h”:

nnnnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh

- O NetId é o conjunto dos bits que representa o endereço da rede.
- O HostId é o conjunto dos bits que representa a máquina dentro da rede.

Por convenção, o endereço de rede possui o campo “HostId” com todos os bits iguais a 0 (zero). Exemplo: considerando que o NetId possui 24 bits e o HostId 8 bits, os endereços IP 192.168.0.10, 192.168.0.30 e 192.168.0.50 pertencem a máquinas da mesma rede 192.168.0.0. O protocolo IP foi definido em classes A, B, C, D e E, que definem diferentes quantidades de bits para o NetId e HostId.

O processo de endereçamento em classe também é conhecido como classful.

Classe	Endereço	Faixa de endereço
A	0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh	1.0.0.0 a 127.255.255.255, sendo para uso privado: 10.0.0.0 a 10.255.255.255
B	10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh	128.0.0.0 a 191.255.255.255, sendo para uso privado: 172.16.0.0 a 172.31.0.0
C	110nnnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh	192.0.0.0 a 223.255.255.255, sendo para uso privado: 192.168.0.0 a 192.168.255.0
D	1110xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	224.0.0.0 a 239.255.255.255
E	1111xxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx	240.0.0.0 a 255.255.255.255

Tabela 1.1
Endereçamento classful.

Na tabela anterior podemos verificar que:

- A classe A possui 8 bits para o NetId e 24 para o HostId, e todo endereço dessa classe inicia com o bit 0;
- A classe B possui 16 bits para o NetId e 16 bits para o HostId, e todo endereço dessa classe começa com os bits 10;
- A classe C possui 24 bits para o NetId e 8 bits para o HostId, e todo endereço dessa classe inicia com os bits 110;
- A classe D é para uso com multicast;
- A classe E foi reservada para uso futuro.

Máscaras de rede

Identificação dos bits NetId e HostId.

- Bits 1 representam NetId.
- Bits 0 representam HostId.
  - 255.255.255.0 = 11111111.11111111.11111111.00000000

Como seria a máscara para uma rede classe B?

Endereços especiais.

- Localhost.
  - Classe A reservada (127.0.0.0).
    - Exemplo: 127.0.0.1
- Endereço de rede.
  - Todos os bits do HostId setados em 0.
    - Exemplo: 130.239.0.0

Endereços especiais

- **Localhost (ou endereço de loopback):** existe em toda máquina que suporta a família de protocolos TCP/IP. Utiliza um endereço de rede classe A reservado (127.0.0.0), do qual é usado apenas o endereço 127.0.0.1. Ele permite a comunicação local (até mesmo um telnet ou FTP para a própria máquina) e a utilização de programas de rede sem estar, necessariamente, conectado a uma rede. Seu uso é ideal para desenvolvimento e testes;
- **Endereço de rede:** composto por todos os bits do HostId iguais a 0 (zero), serve para identificar de forma unívoca uma determinada rede. Exemplo de um endereço de rede classe B: 130.239.0.0.

Endereço de broadcast:

- Todos os bits do HostId setados em 1.
  - Exemplo: 130.239.255.255

Rota default:

- Geralmente representada pelo endereço 0.0.0.0
- **Broadcast:** possibilita o envio de um determinado pacote a todas as máquinas conectadas à rede. O endereço de broadcast é constituído pelo endereço de rede e por todos os bits do HostId iguais a 1 (veremos mais detalhes no item seguinte, sub-redes). Exemplo de um endereço de broadcast em uma rede classe B: rede 130.239.0.0; endereço de broadcast 130.239.255.255 (255.255 = 11111111. 11111111);
- **Rota default**: endereço para aonde devem ser enviados os pacotes cujos destinos não são conhecidos localmente. Esse endereço não pode ser alocado a redes ou máquinas. Muitas vezes a rota default de um host é representada pelo endereço 0.0.0.0.

Alocação baseada em classes gerou problemas.

- Exemplo: classe B para uma rede com 300 hosts.

Soluções:

- Proxy ARP.
- Sub-redes.
- Classless Inter Domain Routing (CIDR).

O padrão Berkeley Software Distribution (BSD) 4.2 fez com que o endereço de broadcast fosse 0, e não 1, para os bits das máquinas. Isso foi desfeito a partir da versão 4.3, por problemas de compatibilidade com os protocolos existentes. Roteadores, switches e placas multi-seriais têm opções para configurar o tipo de broadcast que deve ser feito para haver compatibilidade. Essa opção geralmente é chamada de "zero for broadcast".

A alocação de endereços baseada em classes (classful) gerou muitos problemas de desperdício de endereços. Por exemplo, um provedor que deseja alocar apenas alguns endereços para um pequeno cliente é obrigado a ceder pelo menos 256 endereços de uma classe C. Por outro lado, para um provedor de grande porte (com mais de 256 máquinas), deve-se alocar um endereço classe B com 216 endereços. Para minimizar o desperdício e solucionar esses problemas, foram criados alguns mecanismos, como Proxy ARP, sub-redes e CIDR, que veremos a seguir.

Sub-redes

- Divisão dos endereços de uma classe em subconjuntos.
- Utilização de bits HostId como NetId.
  - Exemplo: dividir uma classe C em 2.
    - 192.168.0.0 (8 bits HostId).
    - 192.168.0.0 – 192.168.0.127 (7 bits Host ID).
    - 192.168.0.128 – 192.168.0.255 (7 bits Host ID).

Uma sub-rede é uma rede física cujos endereços são um subconjunto do conjunto de endereços de uma rede de classe A, B ou C.

Um endereço de sub-rede é obtido através da extensão dos bits do NetId usando bits do HostId do endereço de rede original. Esse processo divide o endereço de rede original em diversos endereços de sub-rede. Com essa solução, é necessário um roteador para conectar as diversas sub-redes criadas.

Para fazer essa quebra são utilizados alguns bits do HostId como sendo do NetId. Exemplo de empréstimo de 8 bits para representar as sub-redes (representada pela letra "s") de uma classe B:

```
nnnnnnnn.nnnnnnnn.ssssssss.hhhhhhhh
```

Nesse processo é necessário utilizar o conceito de máscara de rede (netmask), que indicará quantos bits são usados no NetId estendido e quantos serão usados no HostId.

A máscara de rede é representada por 32 bits com a seguinte propriedade: os bits que representam a sub-rede (o novo NetId) possuem o valor 1 (um) e os bits do HostId possuem o valor 0 (zero). Assim, endereços de rede classe A, B e C equivalem a máscaras de rede 255.0.0.0, 255.255.0.0 e 255.255.255.0, respectivamente. Tais máscaras são denominadas máscaras default. Por exemplo:

Dado o endereço de rede classe B (máscara default 255.255.0.0) 174.179.0.0, pode-se estender o NetId de 16 para 24 bits, ou seja, 8 bits usando a máscara 255.255.255.0 e criando 256 (2^8) sub-redes (174.179.0.0 até 174.179.255.0).

O intervalo de endereços válidos para os hosts da terceira sub-rede 174.179.2.0 255.255.255.0 vai de 174.179.2.1 até 174.179.2.254. O endereço 174.179.2.0 será utilizado para identificar essa sub-rede e o endereço 174.179.2.255 será usado para o endereço de broadcast.

```
10001100.10110011.00000010.00000001     140.179.2.1
(endereço IP classe B)

11111111.11111111.00000000.00000000     255.255.0.0
(máscara default classe B)

11111111.11111111.11111111.00000000     255.255.255.0
(máscara de sub-rede)

-------------------------------------------------

10001100.10110011.00000010.00000000     140.179.2.0
(endereço da sub-rede)

10001100.10110011.00000010.11111111     140.179.2.255
(broadcast da sub-rede)
```

Para entendermos melhor o endereço de rede e o de broadcast, vamos analisar o cálculo em binário. Utilizando a operação AND de um endereço IP com a sua máscara de sub-rede, é possível determinar o endereço da sub-rede. E o endereço de broadcast é o último endereço da sub-rede, ou seja, todos os bits da sub-rede **setados** em 1.

Setado
Anglicismo do verbo "to set", que significa configurar, atribuir um valor.

Exemplo:

- Uma empresa recebeu o endereço de rede classe C 200.211.230.0. Supondo que a empresa tem três escritórios separados e que cada escritório terá, no máximo, 60 máquinas, vamos criar um esquema de sub-redes que atenda a tais condições.

Por meio de um exemplo será mais simples compreender seu uso. Tomemos o caso de uma empresa que recebeu o endereço de rede classe C 200.211.230.0. Supondo que a empresa tem três escritórios separados e que cada escritório terá, no máximo, 60 máquinas, vamos criar um esquema de sub-redes que atenda a tais condições:

- É certo que temos o endereço de rede classe C 200.211.230.0;
- Existe a necessidade de dividir essa rede em três sub-redes, cada uma com no máximo 60 máquinas;
- Para endereçar 60 máquinas é preciso de, no mínimo, 6 bits no campo do HostId (26 = 64);
- Estende-se os 24 bits iniciais do NetId do endereço de rede classe C para 26 bits (32 - 6) e se obtém a máscara de rede 255.255.255.192;
- Distribui-se os endereços nas diferentes sub-redes criadas, como na tabela a seguir:

Rede	Endereço de rede	Broadcast	Netmask	Quantidade de máquinas
Sub-rede 1	200.211.230.0	200.211.230.63	255.255.255.192	62
Sub-rede 2	200.211.230.64	200.211.230.127	255.255.255.192	62
Sub-rede 3	200.211.230.128	200.211.230.191	255.255.255.192	62
Sub-rede 4	200.211.230.192	200.211.230.255	255.255.255.192	62

Tabela 1.2
Sub-redes para atender a empresa com três escritórios separados.

Dessa forma, pode-se criar sub-redes que suportam até 62 máquinas (note que sempre se perdem os endereços das bordas, um para o endereço de sub-rede e outro para o endereço de broadcast). Nesse esquema quatro sub-redes foram definidas. Se forem necessárias apenas três sub-redes, uma delas pode ficar reservada para uso futuro da empresa.

Além disso, para um host de uma sub-rede ter comunicação com outro host de outra sub-rede, será necessário um roteador para atingir todas as sub-redes criadas.

A máscara de rede deve ser igual para todas as sub-redes. No entanto, sistemas com suporte Variable Length Subnet Mask (VLSM) permitem a utilização de máscaras de tamanho variável.

A tabela de NetId/NetMask, apresentada a seguir, resume os valores da máscara de rede (netmask) e os bits de NetID e HostID para diferentes quantidades de máquinas por rede, quando é usado o conceito de sub-rede. Note que à medida que se aumenta o número de sub-redes, o endereço de rede vai progressivamente se estendendo para dentro do campo

de HostID, e consequentemente o número máximo de máquinas por rede vai diminuindo. Por sua vez, os últimos 8 bits da máscara de rede refletem exatamente o número de bits acrescentados ao endereço de rede.

Dado um endereço de rede (ou sub-rede) e sua respectiva máscara, sempre é possível identificar o endereço de broadcast e o número de máquinas suportadas dentro dessa rede.

Endereços/rede	Netmask	NetId/HostID
64	255.255.255.192	11000000
32	255.255.255.224	11100000
16	255.255.255.240	11110000
8	255.255.255.248	11111000

Razões para a adoção do Classless Inter Domain Routing (CIDR):

- Exaustão de classes B.
- Explosão da tabela de roteamento.
- Possível exaustão de endereços.

Com o crescimento da internet e o uso não escalonável da alocação em classe (classful), surgiram sérios problemas de endereçamento, tais como:

- **Exaustão de endereços classes B**: a qualquer instituição que necessitasse mais do que 254 endereços IP, era fornecido um endereço classe B, no qual é possível endereçar até 65.533 hosts. Em alguns casos ainda era esbanjada uma classe A, o que significa dizer que em uma única rede era possível alocar 16 milhões de endereços;
- **Explosão da tabela de roteamento:** como cada classe alocada era uma entrada na tabela de roteamento, e não havia um método de agregar endereços, as tabelas de roteamento se tornaram muito grandes e os roteadores mais sobrecarregados;
- **Possível exaustão de endereços:** como o método por classe não era otimizado para funcionar com redes de qualquer tamanho, havia a possibilidade de não existirem mais classes B ou A para alocação.

A arquitetura CIDR aplicada não impõe que todos os domínios na internet entendam CIDR, pois assume que alguns domínios nunca serão convertidos. Porém, requer que todos os domínios fornecedores de serviço de backbone entendam por completo CIDR.

Atualmente, as alocações de endereço são gerenciadas pela Internet Corporation For Assigned Names and Numbers (ICANN), que delega os processos de alocação de recursos, incluindo o IP, para os Regional Internet Registry (RIR). Em 2002, o ICANN informou um novo RIR para a América Latina e Caribe, chamado Latin American and Caribbean Internet Addresses Registry (LACNIC). Antes disso, essa região era delegada ao American Registry for Internet Numbers (ARIN).

CIDR

- Blocos de endereços e não classes.

Notação:

- Endereço de rede/número de bits do NetId.
  - Exemplo: 192.168.0.0/24
  - Como seria representada a classe A 127.0.0.0?

Por causa desses problemas, foi possível perceber que os endereços poderiam ser conservados se o sistema de classe fosse eliminado. Com isso, um novo esquema chamado Classless Inter Domain Routing (CIDR) passou a adotar medidas como reestruturação da alocação de endereços classe C e agregação hierárquica de endereços. Uma notação CIDR possui informações sobre o número de bits usados para o NetId; por exemplo, 192.168.0.0/22 utiliza 22 bits para o NetId, restando 10 bits para o HostId. Da mesma forma que em sub-redes, cada rede em notação CIDR possui um endereço de rede (bits do HostId todos setados em 0) e o endereço de broadcast (bits do HostId todos setados em 1).

Os endereços de nível mais alto da hierarquia foram divididos pelas regiões geográficas:

- **Europa**: endereços de 194.0.0.0 a 195.255.255.255;
- **América do Norte**: endereços de 198.0.0.0 a 199.255.255.255;
- **América do Sul:** endereços de 200.0.0.0 a 201.255.255.255;
- **Ásia e Pacífico:** endereços de 202.0.0.0 a 203.255.255.255.

Após receber um bloco de endereço, um operador de backbone pode assinalar blocos para outras organizações e assim sucessivamente.

Hoje se observa que muitas vezes a notação de máscara utilizada no CIDR é a mesma utilizada em sub-redes, porque ambas representam a mesma coisa. Exemplo:

```
200.245.120.0/255.255.255.252 = 200.245.120.0/30
```

Conexões ponto-a-ponto para a ligação de dois hosts necessitam apenas de dois endereços IP.

Sendo o endereço 200.238.128.0/24 alocado por um provedor de internet, podemos dividir esse endereçamento para ser utilizado no maior número possível de conexões ponto-a-ponto.

Como necessitamos de dois endereços IP e sabemos que há ainda o endereço de rede e o de broadcast, serão necessários dois bits para o HostId, ou seja, usaremos a máscara /30 ou ainda 255.255.255.252.

Portanto, com um /24 (256 endereços) podemos criar 64 redes/30 (64 x 4 = 256), e com isso conseguiremos endereçar 64 conexões ponto-a-ponto.

Nos sistemas Unix, que compreendem muito bem esses tipos de notação, podemos verificar o endereço de um dispositivo de um host com o comando *ifconfig*:

```
# ifconfig

eth0       Encapsulamento do Link: Ethernet  Endereço de HW
00:0C:29:95:B4:B7

          inet end.: 10.211.8.2  Bcast:10.211.255.255
Masc:255.255.0.0
```

```
          endereço inet6: fe80::20c:29ff:fe95:b4b7/64 Escopo:Link
          UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
          RX packets:14770 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7412 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:1000
          RX bytes:7580861 (7.2 MiB)  TX bytes:484731 (473.3 KiB)
          IRQ:177 Endereço de E/S:0x1080
lo        Encapsulamento do Link: Loopback Local
          inet end.: 127.0.0.1  Masc:255.0.0.0
          endereço inet6: ::1/128 Escopo:Máquina
          UP LOOPBACKRUNNING  MTU:16436  Métrica:1
          RX packets:1231 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1231 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:0
          RX bytes:103910 (101.4 KiB)  TX bytes:103910 (101.4 KiB)
```

Existem outros comandos que listam o endereço dos dispositivos e que representam a máscara no formato CIDR, como o comando *ip*:

```
# ip addr show
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast
qlen 1000
    link/ether 00:0c:29:95:b4:b7 brd ff:ff:ff:ff:ff:ff
    inet 10.211.8.2/16 brd 10.211.255.255 scope global eth0
    inet6 fe80::20c:29ff:fe95:b4b7/64 scope link
       valid_lft forever preferred_lft forever
3: sit0: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
```

Resolução de endereços

- Endereço físico x endereço IP.
- Protocolo ARP.
- Dois hosts na mesma rede (IP).
  - Como eles descobrem o endereço físico um do outro?
  - Broadcast.
  - Qual é o endereço físico do 192.168.0.3?
- Outros hosts podem "ouvir" a resposta e adicionar os endereços físicos x IP em sua tabelas.

A resolução de um endereço IP para o seu respectivo endereço físico é realizada pelo protocolo Address Resolution Protocol (ARP), que associa endereços IP e endereços físicos (MAC, no caso da tecnologia Ethernet) em tabelas mantidas no kernel do sistema.

Para entendermos como isso funciona, imagine que o host A sabe o endereço IP do host B que está na sua própria rede. Como A descobre o endereço físico de B para fazer uma comunicação direta?

Uma solução seria fazer o mapeamento direto, atribuindo o endereço físico do host B junto ao seu endereço IP. Mas isso se torna inviável, já que o endereço Ethernet possui 48 bits e um IP classe C tem apenas 8 bits para representar o host. Para contornar essa situação, o mapeamento é feito de forma dinâmica, onde uma requisição é enviada para o endereço de broadcast solicitando o endereço MAC de um determinado IP. Dessa forma todos os hosts da rede vão capturar a mensagem e apenas o host que possui o endereço IP da requisição informará a todos o seu endereço MAC. As informações de IP x MAC são mantidas pelo Sistema Operacional em uma tabela, e é possível tanto consultá-la como modificar alguns de seus valores.

Existe ainda o Reverse Address Resolution Protocol (RARP), que realiza a resolução inversa, ou seja, resolve um endereço físico para o endereço IP correspondente. Esses protocolos são automatizados no Sistema Operacional e não exigem configurações por parte do administrador, exceto em caso de problemas.

Para visualizar a tabela de resolução de endereços mantida pelo protocolo ARP, em sistemas Unix utiliza-se o comando *arp*:

```
# arp -a

galaxie.pop-pr.rnp.br (192.168.128.30) at 02:60:8C:F1:FF:CD [ether]
on eth0maverick.pop-pr.rnp.br (192.168.128.3) at 00:02:55:5D:07:22
[ether] on eth0
```

Atribuição de endereços (IP)

- Estática.
- Dinâmica:
  - Bootp.
  - DHCP.
  - PPP.

Existem duas formas de se atribuir um endereço IP a uma interface de rede: estática ou dinamicamente.

Na atribuição estática, o administrador da máquina configura manualmente um endereço IP a ser atribuído a determinada interface. No caso específico de sistemas Unix, essa atribuição é feita em arquivos de configuração que são lidos por scripts executados na inicialização do sistema.

Já na atribuição dinâmica são usados protocolos especiais que permitem que a máquina solicite a um servidor o endereço IP que ela deverá usar. Existem alguns protocolos para isso, sendo os mais comuns:

- **Bootp**: apesar de antigo, ainda é utilizado. Quando ligada, uma máquina transmite um broadcast para a rede (lembre-se de que ela passa o seu endereço MAC de origem no quadro Ethernet), requisitando o seu endereço IP. Baseado no endereço MAC recebido no pacote, o servidor bootp identifica o endereço IP reservado para a máquina, consultando uma tabela que associa endereços IP e endereços MAC. Em seguida, o servidor envia uma resposta, também para o endereço de broadcast, indicando o endereço IP da máquina. Esse protocolo é utilizado principalmente por máquinas sem disco (diskless) que realizam boot através da rede;
- **Dynamic Host Configuration Protocol (DHCP):** semelhante ao bootp, porém aloca endereços dinamicamente a partir de um bloco de endereços especificados pelo administrador (uma máquina pode receber um endereço diferente a cada boot). Esse protocolo simplifica o processo de atribuição de endereços, permitindo a configuração de uma grande rede de forma centralizada. Também permite a configuração de outros parâmetros de rede além do endereço IP das máquinas;
- **PPP**: esse protocolo não faz somente a alocação de endereços, mas também é um protocolo de comunicação entre dois pontos. A atribuição dinâmica de endereços é apenas uma das muitas propriedades desse serviço. Após estabelecer a conexão, o PPP dinamicamente seleciona um endereço livre e o atribui para a máquina remota conectada via um modem. É bastante usado por provedores de acesso;
- **RARP:** também utilizado para a atribuição dinâmica de endereço. No entanto, é um protocolo em desuso que tem como principais desvantagens ser implementado na camada 2 (dependente, portanto, da tecnologia usada nessa camada) além de só permitir a obtenção de endereço IP e nada mais (DNS, rotas etc.).

Roteiro de Atividades 1

Atividade 1.1 – Entendendo o modelo ISO/OSI

O diretor de uma empresa deseja fechar um negócio com o diretor de outra empresa. Mostre como essa negociação poderia ser representada no modelo OSI, usando três camadas (telefonista, secretária e diretor). Suponha que o meio de comunicação a ser usado é o fax, que a atribuição da secretária é lidar com nomes de pessoas e a da telefonista é lidar com números de telefone.

Atividade 1.2 – Visualizando a configuração de rede em uma máquina Linux

Utilize o comando *ifconfig* sem argumentos. Caso o caminho para o *ifconfig* não esteja na sua variável de ambiente *path*, será necessário digitar */sbin/ifconfig*. A partir da saída desse comando, identifique:

1. Qual a tecnologia de nível 2 que está sendo utilizada?
2. Qual o endereço físico (MAC) da máquina?
3. Qual o endereço IP e a máscara de rede?
4. É possível obter alguma informação sobre o nível 1?

Atividade 1.3 – Entendendo a tabela ARP

Nesta atividade, vamos explorar as informações da tabela ARP:

1. Verifique o número de entradas da tabela ARP da sua estação de trabalho utilizando o comando *arp*.
2. Efetue um *ping* para uma estação na mesma rede que a sua.
3. Examine de novo a tabela ARP e compare com o resultado anterior. O que aconteceu?
4. Qual foi o processo envolvido para entregar o pacote até o host de destino?
5. Que mapeamentos foram feitos entre o IP e o MAC Address?
6. As próximas conexões vão seguir o mesmo processo?

Atividade 1.4 – Cabeçalho Ethernet

No cabeçalho de um quadro Ethernet o endereço de destino precede o endereço de origem. Você consegue ver alguma vantagem nisso? Qual?

Atividade 1.5 – Fragmentação

Suponha que uma determinada tecnologia de nível 2 consiga transmitir quadros de 800 bytes (incluído o cabeçalho).

O que aconteceria ao tentar transmitir um datagrama IP de 1800 bytes?

Atividade 1.6 – Criando endereços de rede ou sub-rede

Um provedor de acesso alocou para um cliente uma sub-rede de 16 endereços, com endereço de rede 200.231.15.240. Quais são os endereços de rede, broadcast, máscara de rede e os endereços válidos para esse cliente?

Atividade 1.7 – Notação CIDR

A notação de rede e a máscara de rede com notação CIDR também podem ser usadas para referenciar uma única máquina. Como isso seria representado? As regras de endereços de rede e de broadcast continuam valendo nessa notação?

Atividade 1.8 – Faixa de endereços válidos na internet

Quais faixas de IP abaixo são consideradas públicas, isto é, quais podem ser usadas para fazer uma ligação na internet? Considere as máscaras default para cada classe e explique a resposta.

1. 200.249.30.0
2. 10.0.3.0
3. 127.1.2.0
4. 200.245.20.0
5. 200.261.242.0
6. 200.255.254.0
7. 2.0.0.0
8. 167.198.2.255

Atividade 1.9 – Entendendo a máscara de rede

Preencha os campos abaixo com a máscara equivalente e cite a quantidade de hosts que se pode ter. Não considere os endereços utilizados para identificar a rede e o broadcast da rede. Exemplo:

255.255.255.0 = /24 = 28 = 256

1. 255.255.255.255 = _____ = _____
2. 255.255.255.128 = _____ = _____
3. _____ = /28 = _____ = _____
4. _____ = /20 = _____ = _____
5. 255.255.0.0 = _____ = _____
6. 255.248.0.0 = _____ = _____
7. _____ = /30 = _____ = _____

Atividade 1.10 – Endereçamento dinâmico

É possível obter um endereço IP dinamicamente, em vez de configurá-lo previamente no Sistema Operacional. No entanto, como fazer para obter um endereço IP tendo que se comunicar em uma rede IP e sem ter ainda um endereço IP?

Atividade 1.11 – Resolução de nomes

1. Efetue um *ping* para o endereço www.esr.rnp.br. Qual foi o endereço IP obtido?
2. O que aconteceria se incluíssemos uma linha "127.0.0.1 www.esr.rnp.br" no arquivo */etc/hosts*?

2

Introdução à administração de redes e arquitetura TCP/IP (parte 2)

objetivos

Entender como funciona o mecanismo de roteamento e o protocolo de controle TCP/IP, o princípio de funcionamento dos protocolos TCP e UDP, bem como a diferença entre eles, e a arquitetura cliente/servidor e suas principais vantagens.

conceitos

Mecanismo de roteamento e protocolo de controle TCP/IP, protocolos TCP e UDP e arquitetura cliente/servidor.

Roteamento

Processo de escolha das rotas para alcançar um determinado destino.

- A internet existiria se não houvesse roteamento?

Hosts multi-homed:

- Conectam-se a duas redes diferentes, mas não encaminham pacotes IP.
- Hosts x hosts multi-homed x roteadores.

Encaminhamento direto x encaminhamento indireto.

O roteamento é o processo de escolha das rotas que os datagramas devem seguir para alcançar o seu destino. Neste Capítulo, trataremos do roteamento IP.

Roteadores e hosts multi-homed

Nesse sentido, hosts que se conectam a duas redes diferentes, mas que não desempenham esse papel de encaminhamento de pacotes IP, são chamados de hosts multi-homed.

Encaminhamento direto e indireto

Imagine um host conectado a uma rede local. Ao transmitir seus pacotes, o Sistema Operacional verifica se o IP de destino pertence ou não à mesma rede física. No primeiro caso, basta transmitir os pacotes diretamente para a máquina destino. Essa situação é denominada encaminhamento direto. Se o destino não estiver na mesma rede física, o transmissor deve passar os pacotes para um roteador encarregado de encaminhá-los ao destinatário. Esse caso é denominado encaminhamento indireto.

Tabela de roteamento

Relaciona destinos a rotas.

Próximo passo (next hop).

- Próximo roteador no caminho até o destino final.

Tipos de rotas:

- Para host.
- Para rede.

Default.Como uma máquina decide para quem encaminhar um pacote?

O roteamento IP utiliza o que se denomina de tabela de roteamento. Cada máquina e cada roteador possuem pelo menos uma tabela de roteamento IP. Essa tabela contém possíveis destinos e modos para acessá-los. Na tabela de roteamento podemos encontrar três tipos de rotas:

- Rotas para um host;
- Rotas para uma rede;
- Rota default.

A rota default é um tipo de rota especial que significa todos os destinos possíveis, e é normalmente associada ao endereço 0.0.0.0 com máscara 0.0.0.0. Note que seria praticamente impossível manter na tabela de roteamento todos os destinos possíveis na internet, assim como também seria bastante difícil incluir todo o caminho necessário para atingir um determinado destino. A utilização da rota default minimiza bastante o primeiro problema.

Próximo passo (next-hop)

Conforme observado no item anterior, seria praticamente impossível manter nas tabelas de roteamento todo o caminho para um determinado destino. Como os roteadores trabalham de forma colaborativa, basta que na tabela tenhamos o endereço do próximo passo (next-hop), ou seja, do próximo roteador no caminho até o destino final.

Exemplo de topologia

Como seria a tabela de roteamento de R1?

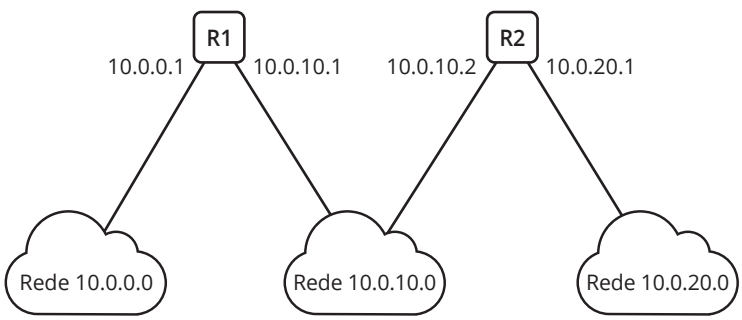


Figura 2.1
Três redes interconectadas pelos roteadores R1 e R2.

Considere a imagem anterior: nesse exemplo, temos três redes interconectadas pelos roteadores R1 e R2. Cada roteador possui duas interfaces e os seus endereços estão listados ao lado dos roteadores. Vamos construir a tabela de roteamento de R1, de modo que ele saiba como atingir todas as redes:

Destino	Próximo passo
Rede 10.0.0.0/24	Entrega direta
Rede 10.0.10.0/24	Entrega direta
Rede 10.0.20.0/24	10.0.10.2

Tabela 2.1 Tabela de roteamento de R1.

Vamos construir também a tabela de rotas para um host situado na rede 10.0.0.0:

Destino	Próximo passo
Rede 10.0.0.0/24	Entrega direta
0.0.0.0	10.0.0.1

Tabela 2.2 Host situado na rede 10.0.0.0.

Observe que esse host não sabe exatamente como chegar à rede 10.0.20.0, mas possui uma rota default que aponta para o roteador R1, ou seja, para atingir um destino que não esteja na rede 10.0.0.0, basta encaminhar os pacotes para o roteador R1.

Rotas estáticas x rotas dinâmicas

Estáticas: configuradas pelo administrador.

Dinâmicas: protocolos de roteamento.

- Exemplo: Open Shortest Path First (OSPF).
- Exemplo: Border Gateway Protocol (BGP).

Como evitar loops?

- Time To Live (TTL).
- Internet Control Message Protocol (ICMP).

A tabela de roteamento de um host ou roteador pode ser preenchida de duas formas:

- As rotas podem ser configuradas manualmente pelo administrador;
- Pode-se executar um aplicativo que, com o auxílio de um protocolo de roteamento, vai preencher automaticamente essa tabela.

O primeiro caso chamamos de roteamento estático e é o tipo de roteamento mais apropriado para pequenas redes e hosts em geral. No entanto, nos roteadores de redes médias e grandes, administrar uma grande quantidade de rotas manualmente seria um processo bastante complicado e sujeito a erros. Para esses casos são utilizados protocolos de roteamento, como o Open Shortest Path First (OSPF) e o Border Gateway Protocol (BGP), para permitir que os roteadores se comuniquem trocando informações sobre rotas e destinos.

Tempo de vida (TTL)

Volte ao exemplo do item "Próximo passo" e imagine que, por alguma razão, o roteador R2 está com sua tabela de roteamento configurada de maneira errada. Em vez de fazer o encaminhamento direto para a rede 10.0.20.0/24, seu próximo passo (next-hop) para essa rede é o endereço 10.0.10.1 do roteador R1. O que aconteceria com um pacote IP que saísse de um host da rede 10.0.0.0/24 com destino a um host da rede 10.0.20.0/24? Esse pacote entraria em um loop infinito entre os roteadores R1 e R2, já que cada um aponta para o outro como o próximo passo (next-hop) para a rede 10.0.20.0/24. Como visto no Capítulo 1, no cabeçalho

(header) do IP existe um campo chamado “Time To Live (TTL)”. Esse campo contém um valor que é decrementado a cada passo dado por um pacote IP na rede. Ou seja, cada vez que ele atravessa um roteador, esse valor é subtraído de 1 unidade. Ao chegar a 0, o pacote deve ser descartado, mesmo que ele não tenha chegado ao seu destino final, e um pacote ICMP contendo o erro é enviado ao remetente.

Algoritmo de roteamento

- Se o destino é diretamente conectado: encaminhamento direto.
- Se existe rota para o host: use a rota.
- Se existe rota para a rede: use a rota.
- Se existe rota default: use a rota.
- Do contrário: erro de roteamento.

De forma resumida, podemos descrever o algoritmo de roteamento da seguinte forma:

- Se o destino pertence a alguma rede diretamente conectada, enviar o pacote via interface de rede diretamente conectada;
- Se existe uma rota de host para o destino, enviar o pacote para o roteador indicado como próximo passo (next-hop);
- Se existe uma rota de rede para o destino, enviar o pacote para o roteador indicado como próximo passo (next-hop);
- Se existe uma rota default, enviar o pacote para o roteador indicado na rota default;
- Do contrário, ocorre erro de roteamento.

Rotas mais específicas têm precedência. No caso de existirem duas ou mais rotas de redes para um mesmo host, “vence” a que possuir a maior máscara. Por exemplo, no caso de uma rota para a rede 10.0.0.0/24 e de outra para a rede 10.0.0.0/18, para atingir o host 10.0.0.1 a rota a ser tomada é a que aponta para 10.0.0.0/24.

Internet Control Message Protocol (ICMP)

- Utiliza o IP.
- Opera no nível 3.

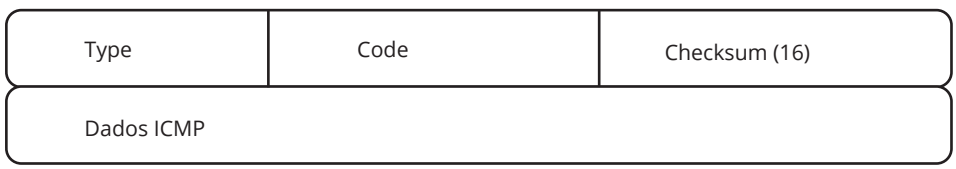


Figura 2.2
Exemplo de cabeçalho ICMP.

Internet Control Message Protocol (ICMP) é um protocolo utilizado por todos os hosts TCP/IP para trocarem informações de controle e erro. Apesar de mensagens ICMP serem encapsuladas dentro de datagramas IP, o ICMP não é um protocolo de transporte, fazendo parte, portanto, da camada de rede. Uma mensagem ICMP é transmitida na parte de dados do pacote IP.

O protocolo ICMP utiliza os seguintes tipos de mensagens:

Tipo	Descrição
0	echo reply
3	destination unreachable
4	source quench
5	redirect
8	echo request
9	router advertisement
10	route solicitation
11	time exceeded
12	parameter problem
13	timestamp
14	timestamp reply
15	information request
16	information reply
17	address mask request
18	address mask reply

Tabela 2.3
Tipos de mensagens do protocolo ICMP.

O formato das mensagens ICMP, que são encapsuladas dentro do IP, possui forma variável, de acordo com o tipo da mensagem.

Principais tipos:

- Tipo 0: echo reply.
- Tipo 8: echo request.
- Tipo 3: destination unreachable.
- Tipo 11: time exceed.

Cada tipo pode ter códigos diferentes:

- Tipo 3, código 0: network unreachable.
- Tipo 3, código 1: host unreachable.

Os tipos de mensagens ICMP também possuem um código. Por exemplo, uma mensagem do tipo 3 (destination unreachable) pode ser classificada com o código 0 (network unreachable), código 1 (host unreachable), código 2 (protocol unreachable) e outros códigos. A tabela a seguir resume os tipos e os códigos usados no ICMP:

Type	Code	Descrição
0	0	echo reply
3	0	network unreachable
3	1	host unreachable
3	2	protocol unreachable
3	3	port unreachable
3	4	fragmentation needed but no frag. bit set
3	5	source routing failed
3	6	destination network unknown
3	7	destination host unknown
3	8	source host isolated (obsoleto)
3	9	destination network administratively prohibited
3	10	destination host administratively prohibited
3	11	network unreachable for TOS
3	12	host unreachable for TOS
3	13	communication administratively prohibited by filtering
3	14	host precedence violation
3	15	precedence cutoff in effect
4	0	source quench
5	0	redirect for network
5	1	redirect for host
5	2	redirect for TOS and network
5	3	redirect for TOS and host
8	0	echo request
9	0	router advertisement
10	0	route solicitation
11	0	TTL equals 0 during transit
11	1	TTL equals 0 during reassembly
12	0	IP header bad (catchall error)
12	1	required options missing
13	0	timestamp request (obsoleto)
14	0	timestamp reply (obsoleto)
15	0	information request (obsoleto)

Tabela 2.4
Tipos e códigos do ICMP.

Vários comandos de avaliação de conectividade utilizam mensagens do protocolo ICMP, como o *ping* e o *traceroute*.

Ping

- ICMP tipo 0 e 8.
- Que informações ele fornece?

Como funciona o traceroute?

- Pacotes UDP.
- Manipulação do TTL.
- ICMP tipo 11.
- ICMP tipo 3.

Envia uma mensagem ICMP do tipo 8 (echo request) para outra máquina que, ao recebê-la, retorna uma mensagem ICMP do tipo 0 (echo reply) como resposta. O objetivo é, primeiramente, verificar se uma máquina está no ar. Em adição, os pacotes retornados permitem medir o tempo de ida e volta (Round-Trip-Time – RTT). Exemplo:

```
# ping www.rnp.br

PING www.rnp.br (200.130.25.20) 56(84) bytes of data.

64 bytes from jupiter.sso.rnp.br (200.130.25.20): icmp_seq=1 ttl=57
time=54.0 ms

64 bytes from jupiter.sso.rnp.br (200.130.25.20): icmp_seq=2 ttl=57
time=51.3 ms

64 bytes from jupiter.sso.rnp.br (200.130.25.20): icmp_seq=3 ttl=57
time=51.2 ms

64 bytes from jupiter.sso.rnp.br (200.130.25.20): icmp_seq=4 ttl=57
time=51.1 ms

--- www.rnp.br ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3002ms rtt
min/avg/max/mdev = 51.193/51.973/54.067/1.242 ms
```

Traceroute

Mostra os roteadores intermediários que definem a rota até a máquina destino especificada. Para definir a lista de roteadores, o traceroute utiliza o campo "TTL" do pacote IP, que indica o número máximo de roteadores que o roteador pode passar.

Inicialmente, o traceroute envia um pacote IP setando o TTL para 1. Logo que o pacote IP chega ao primeiro roteador, o TTL expira e o roteador retorna uma mensagem ICMP do tipo 11 (time exceeded). Ao receber a mensagem, a ferramenta descobre o nome do primeiro roteador. Em seguida, é enviado com TTL setado para 2 e assim até atingir o host final.

Porém, o TTL não vai exceder ao chegar ao destino final, mas como esse pacote IP está encapsulando dados UDP direcionados para uma porta, será gerado um ICMP do tipo 3 (destination unreachable), o qual informará que a porta está inatingível para a ferramenta traceroute. Ao receber tal notificação, descobre a lista completa de hosts. Exemplo:

```
# traceroute www.rnp.br

traceroute: Warning: www.rnp.br has multiple addresses; using
200.17.63.100

traceroute to www.rnp.br (200.17.63.100), 30 hops max, 38 byte
packets

1 bb3.pop-pr.rnp.br (200.134.255.1)  0.968 ms  0.597 ms  0.593 ms

2 sp-pos0-0.bb3.rnp.br (200.143.253.106)  9.339 ms  9.017 ms  8.942 ms

3 rj-pos2-0.bb3.rnp.br (200.143.253.102)  14.900 ms  15.076 ms  14.888
ms

4 rj7507-fastethernet6-0.bb3.rnp.br (200.143.254.253)  15.717 ms
15.641 ms

16.044 ms

5 ambrosia-serial0-0.bb3.rnp.br (200.143.255.33) 18.675 ms  16.826 ms
17.317 ms

6 janeway-e.nc-rj.rnp.br (200.17.63.190)  17.528 ms  16.982 ms  16.982 ms

7 ciprod.nc-rj.rnp.br (200.17.63.100)  17.521 ms 17.249 ms  17.330 ms
```

Nível de transporte

Relembrando:

- O protocolo IP é não confiável e não orientado à conexão.
- O protocolo IP só identifica hosts.
  - Como saber para que processo se destina um determinado pacote?

Vimos que o protocolo IP é não confiável e não orientado à conexão. Além disso, o protocolo IP apenas identifica máquinas, mas não identifica usuários e/ou aplicativos. A seguir, veremos dois protocolos do nível de transporte criados para solucionar esses problemas.

Portas

- Porta de origem.
- Porta de destino.

Como é feita a atribuição de portas?

- Atribuição da porta de origem.
- Atribuição de porta de destino.

Os protocolos de transporte implementam os mecanismos necessários para resolver o problema da identificação de um usuário e/ou aplicativo (que aqui chamaremos de processos) em cada host. Essa identificação é feita através da utilização de identificadores de 16 bits, também conhecidos como portas.

Cada cabeçalho de um datagrama de um protocolo de transporte possui:

- Um campo que contém a porta de origem (que identifica o processo transmissor);
- Um campo que contém a porta de destino (que identifica o processo receptor).

E como um processo identifica outro processo em outra máquina? Como é feita a atribuição dos números de porta? Os projetistas dos protocolos de transporte optaram por uma atribuição mista, ou seja, algumas portas são atribuídas por uma autoridade central, enquanto outras podem ser atribuídas dinamicamente pelas aplicações. Os servidores de nome (DNS), por exemplo, utilizam a porta 53. Assim um processo cliente sabe que na porta 53 da máquina servidora de DNS um processo servidor de nomes aceitará datagramas. Nos sistemas Unix ou Linux, o arquivo */etc/services* guarda uma lista de portas "conhecidas".

Nível de transporte: protocolo UDP

- User Datagram Protocol (UDP).
- Não confiável e não orientado à conexão.
- No entanto, possui portas.

Porta de origem	Porta de origem
Comprimento da mensagem UDP	Soma de verificação
Dados	

Figura 2.3
Cabeçalho UDP.

Assim como o IP, o protocolo User Datagram Protocol (UDP), também é um protocolo não confiável e não orientado à conexão. No entanto, utiliza os identificadores de portas, como podemos ver no cabeçalho do exemplo anterior.

Apesar de o protocolo UDP possuir um campo de soma de verificação, sua utilização é opcional, ficando a cargo do programador utilizar ou não esse campo. Vale lembrar também que caso uma aplicação escolha utilizar o protocolo de transporte UDP, fica a cargo da aplicação: o controle de perdas, a ordenação dos pacotes, a retransmissão e a detecção de erros.

O protocolo UDP foi criado para permitir grande flexibilidade ao programador ao custo, é claro, de uma complexidade maior na programação de aplicativos. O processo na máquina transmissora solicita uma porta ao Sistema Operacional para transmitir seus datagramas. Após a alocação da porta, o processo identifica o IP de destino e a porta de destino e envia o datagrama. Não há nenhuma negociação prévia entre o transmissor e o receptor antes do envio. Caso não haja nenhum processo na máquina de destino escutando naquela porta, a máquina de destino gera uma mensagem ICMP porta não atingida (tipo 3, código 3).

Nível de transporte: protocolo TCP

- Usa o IP, mas é confiável e orientado à conexão.
- Sua confiabilidade é possível pelo uso da técnica de confirmação positiva com retransmissão.
- Possui maior complexidade.

O protocolo Transmission Control Protocol (TCP) é um protocolo orientado à conexão e confiável, além de possuir o mesmo mecanismo de portas. Mas como um protocolo pode oferecer conexão e confiabilidade se ele utiliza o IP, que é um protocolo não confiável e não orientado à conexão?

A confiabilidade é implementada no TCP através da técnica de confirmação positiva com retransmissão. Nessa técnica, para cada segmento de dados, o transmissor aguarda do receptor uma mensagem de confirmação (ACK). Para o caso de o segmento não ter alcançado o receptor, ou de a mensagem de confirmação ter sido perdida, o protocolo se utiliza de um conjunto de temporizadores. Quando um temporizador se esgota sem que o transmissor tenha recebido uma confirmação, o segmento para o qual se esperava a confirmação é retransmitido.

Para estabelecer uma conexão, o protocolo TCP utiliza campos especiais do cabeçalho e técnicas de handshake no início e ao final da transmissão de uma conexão (também chamada de fluxo ou stream). Além disso, utilizando-se a tupla (IP de origem, porta de origem, IP de destino, porta de destino), é possível identificar uma conexão. Antes de estudarmos com mais detalhes como se estabelece uma conexão, vamos estudar o cabeçalho do protocolo TCP.

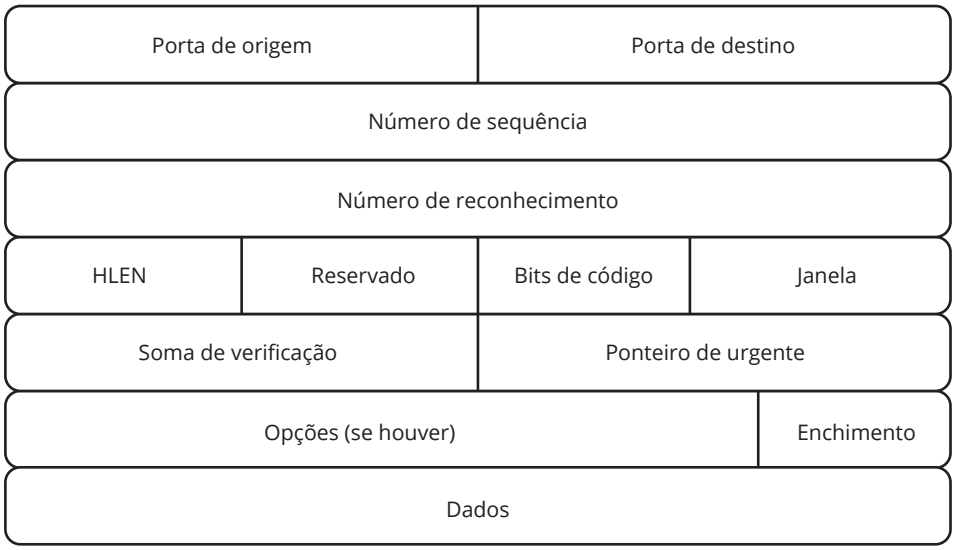


Figura 2.4
Cabeçalho TCP.

Bit	Significado
URG	Segmento que contém dados urgentes.
ACK	Confirmação de recebimento.
PSH	Este segmento requer um push (evita a bufferização).
RST	Abortar a conexão.
SYN	Usado para iniciar uma conexão.
FIN	Usado para finalizar uma conexão.

Tabela 2.5
Bits de reconhecimento.

Início e término de uma conexão

Para iniciar uma conexão é necessário que as duas pontas negociem o estabelecimento dessa conexão. No TCP essa negociação é feita através do chamado 3 way handshake:

- A máquina T envia um datagrama com o bit SYN (localizado no campo “bits de código”) ligado;
- A máquina R recebe esse datagrama e envia de volta um datagrama confirmando o recebimento do datagrama de T, ligando o bit ACK e também solicitando uma abertura de conexão ligando o bit SYN;
- A máquina T recebe esse datagrama e envia uma confirmação, ou seja, mais um datagrama com o bit ACK ligado. A transmissão de dados já pode ser iniciada a partir desse pacote.

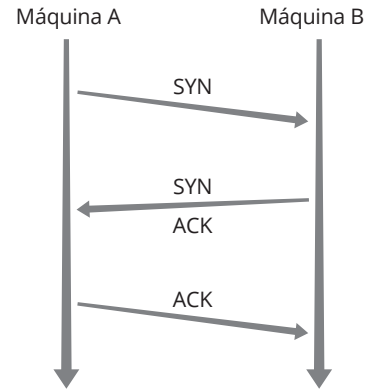


Figura 2.5
Como funciona o 3 way handshake.

Término de uma conexão

O término da conexão ocorre de maneira um pouco diferente. Vejamos como isso acontece:

- A máquina T já enviou tudo o que ela precisava e quer fechar a conexão. Ela envia, então, um datagrama para R com o bit FIN ligado;
- A máquina R envia uma confirmação ACK para esse datagrama. Isso encerra a conexão em um dos sentidos apenas, já que R ainda pode ter dados para enviar a T. Quando R terminar, ela envia um datagrama para T com bit FIN ligado;
- A máquina T envia um ACK de volta e a conexão está encerrada.

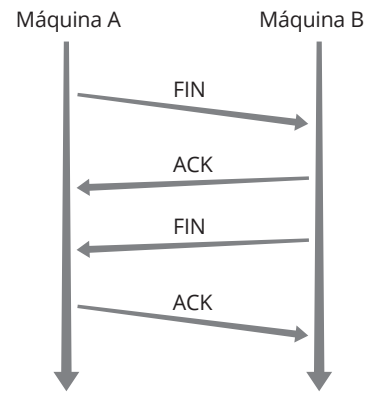


Figura 2.6
Como termina uma conexão.

Nível de transporte: conexão virtual

Conexão:

- (<IP origem:porta origem>, <IP destino:porta destino>)

Lembre-se: a conexão é full duplex.

- Transporta dados nos dois sentidos da conexão.

Utilidade dos campos “números de sequência” e “número de reconhecimento”.

- Identificar os pacotes dentro de uma conexão.

O TCP garante fluxo confiável e serviço de conexão virtual para os aplicativos através de confirmações e, quando necessário, através da retransmissão de pacotes. O TCP não só é capaz de estabelecer conexões, mas também essas conexões operam no modo full duplex, ou seja, são capazes de transportar dados nos dois sentidos da conexão. Além disso, o TCP possui mecanismos para ordenar os pacotes, para o caso de eles chegarem fora de ordem.

Ao estabelecer uma conexão, os pacotes TCP carregam no campo "número de sequência" um identificador do pacote dentro do fluxo. Esse número é incrementado à medida que os dados vão sendo transmitidos e/ou recebidos. No campo "número de reconhecimento", o transmissor envia o próximo número de sequência que ele espera receber do receptor.

Como vimos, para cada segmento recebido é emitida uma mensagem de confirmação. Essa mensagem é, na verdade, apenas um bit marcado no cabeçalho do pacote. Assim as confirmações de recebimento podem "pegar carona" em pacotes de dados.

Muitos detalhes sobre o protocolo TCP foram omitidos aqui. Recomendamos a leitura da bibliografia recomendada para mais informações.

Veja o exemplo:

- O host T vai se comunicar com o host R;
- O host T envia um datagrama para R e o identifica com o número 3500;
- R recebe a mensagem e também envia para T um datagrama, identificando-o com o número 7290 e preenchendo o campo de número de recebimento com o identificador de T + 1.

Cada host utiliza sua própria sequência de identificação, mas a resposta a cada datagrama pode carregar também uma confirmação de recebimento que utiliza a sequência do outro host.

Nível de transporte

- TCP x UDP.
- Se o TCP é tão bom, por que usar o UDP?
- Resolução de nomes (DNS) com UDP.
  - Dois pacotes.
  - Se fosse utilizado TCP, sete pacotes seriam usados só para abrir e fechar a conexão.
- Transferência de arquivos.
  - Se fosse utilizado UDP, como seriam ordenados os pacotes? E quanto às perdas?

A principal diferença entre esses protocolos é que o TCP é orientado à conexão, ou seja, possui vários mecanismos para corrigir e retransmitir pacotes corrompidos ou perdidos. O UDP, por sua vez, não realiza a checagem nem a confirmação de recebimento: os dados são transmitidos uma única vez e os pacotes corrompidos são simplesmente descartados.

Então, se o TCP é tão bom, por que usar o UDP? O problema consiste simplesmente na quantidade de pacotes utilizados pelo TCP para garantir a entrega dos dados. No UDP, como a transmissão é realizada utilizando o maior desempenho possível, uma grande quantidade de informações de controle é eliminada dos pacotes, com enfoque apenas na transmissão dos dados. A correção de erros deve ser realizada pela aplicação que usa esse tipo de protocolo (UDP).

Arquitetura cliente/servidor

Quem é o cliente?

- O que inicia a conexão.

Quem é o servidor?

- O que "escuta" em uma porta e aguarda conexões.

No Unix/Linux, servidores também são chamados de daemons.

O modelo TCP/IP é baseado na arquitetura cliente/servidor. O cliente envia uma requisição

O modelo TCP/IP é baseado no conceito de portas e conexões para assegurar que dados recebidos por uma máquina sejam corretamente repassados aos processos de destino.

ao servidor, que por sua vez processa essa requisição e retorna o resultado ao cliente. Nessa arquitetura, um servidor pode atender, simultaneamente, a diversas requisições de diferentes clientes.

Um servidor oferece seu serviço em uma porta padrão conhecida. Após ser inicializado, o servidor aguarda por requisições na sua porta. Por outro lado, o Sistema Operacional de um cliente escolhe uma porta aleatória não utilizada e solicita uma conexão com a porta padrão do servidor (no caso do TCP) ou envia diretamente pacotes para a porta do servidor (no caso do UDP). Uma conexão identifica de forma única o canal de comunicação estabelecido entre um processo cliente e um processo servidor. A conexão é identificada pelos endereços IP e pelas portas do cliente e do servidor.

Os servidores são implementados através de processos especiais conhecidos como **daemons**. Em geral os daemons são conhecidos pelo nome do serviço acrescido da letra "d".

Daemons

Em Unix e outros sistemas operacionais multitarefas, um daemon, acrônimo de Disk And Execution MONitor (Monitor de Execução e de Disco) é um programa que roda de forma independente em background, em vez de ser controlado diretamente por um usuário.

Principais daemons ou portas:

- 21/tcp: ftp: usado para transferência de arquivos entre máquinas.
- 23/tcp: telnet: serviço de terminal remoto.
- 22/tcp: ssh: permite comunicação segura entre duas máquinas, utilizando mecanismos de criptografia.
- 25/tcp: smtp: serviço de envio e recebimento de e-mails.
- 53/udp: dns: serviço de resolução de nomes.
- 80/tcp: www: servidor web http.
- 110/tcp: pop3: permite leitura e transferência remota de e-mails.
- 443/tcp; https: www seguro, permite transferência de páginas do servidor ao cliente sem que possam ser lidas por terceiros.

Uma vantagem das RFCs é que elas são debatidas em fóruns de discussão que tornam pública a opinião dos profissionais mais importantes nesses assuntos: os implementadores.

Acima podemos ver uma lista dos daemons, protocolos mais conhecidos e suas portas. Os protocolos são especificados em documentos denominados Request for Comments (RFCs), que descrevem os diferentes tipos de mensagens suportados pelos protocolos. A especificação de um protocolo assegura a compatibilidade das diversas implementações.

Algumas portas são padronizadas.

- Exemplo: portas 21, 23, 80 e 110.

No Linux/Unix:

- Portas baixas: 0 a 1023, só o root pode usá-las.
- Portas altas: 1024 a 65535.

As portas no intervalo de 0 a 1023 são conhecidas como portas restritas ou portas baixas. Os serviços que rodam nessas portas precisam ter permissão SUID = 0 (root), o que pode gerar graves consequências caso algum deles tenha qualquer problema de segurança. Atualmente, os programas que usam tais portas procuram iniciar como root, apenas para ganhar acesso à porta e depois mudar o processo para um usuário sem tantos poderes. Por isso existem usuários chamados www, FTP, postmaster, Squid, Samba etc.

As portas de 1024 até 65535 são conhecidas como portas altas e são geralmente as portas aleatórias utilizadas pelos clientes. Como tais portas podem ser abertas por qualquer usuário, tornou-se comum usá-las também para serviços experimentais. No entanto, alguns desses serviços podem ser padronizados posteriormente e continuar a utilizar as portas

altas. Por exemplo: Squid (proxy), ICQ ou VNC.

Um aplicativo cliente/servidor muito conhecido é o Telnet, usado para obter uma sessão interativa com outro host. O protocolo é bastante simples e consiste basicamente no envio de comandos pelo cliente e no envio dos resultados dos comandos pelo servidor. A simplicidade do protocolo permite sua utilização para testar outros protocolos. Por exemplo, pode-se testar um servidor web sem utilizar um browser. Para isso, basta saber os tipos de mensagens do protocolo a ser testado.

A seguir, dois exemplos: no primeiro, o Telnet é usado para leitura de e-mails, via Pop; no segundo, para uma requisição de página a um servidor web:

Exemplo 1:

```
# telnet 10.4.0.1 110
.Trying 10.4.0.1... Connected to 10.4.0.1. Escape character is '^]'.
+OK QPOP (version 2.53) at localhost starting. USER cristina
+OK Password required for cristina. PASS supersenha
+OK cristina has 2 messages (8614 octets). LIST
+OK 2 messages (8614 octets)
1 886
2 7728
DELE 1
+OK Message 1 has been deleted. QUIT
```

Exemplo 2:

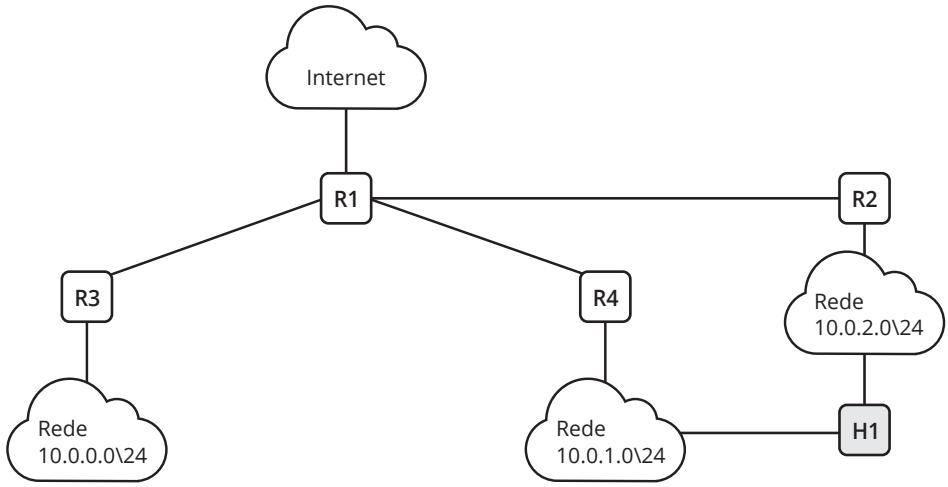
```
# telnet www.rnp.br 80
Trying 200.130.25.20...
Connected to jupiter.sso.rnp.br. Escape character is '^]'.
GET /index.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
...
Connection closed by foreign host.
```

Os comandos digitados dentro do Telnet (em destaque) são as mensagens específicas dos protocolos. Os clientes oferecem uma interface com o usuário e, internamente, trocam mensagens do protocolo específico com o servidor. A padronização dos protocolos viabiliza a existência de diversas implementações de clientes e servidores. Por exemplo, web browsers e servidores de e-mail. Dessa forma, os usuários têm total liberdade para escolher as implementações que melhor atendam às suas necessidades.

Roteiro de Atividades 2

Atividade 2.1 – Roteamento

Considere a seguinte topologia:



1. Atribua endereços às interfaces de conexão entre os roteadores R1, R2, R3 e R4, com prefixo 192.168.0.X. Utilize a menor máscara de rede possível e atribua endereços para as interfaces das redes 10.0.X.0/24.
2. Construa a tabela de rotas dos roteadores R1, R2, R3, R4 e do host multi-homed H1; também atribua endereços para as suas interfaces.
3. Construa a tabela de rotas do host 10.0.0.34.
4. Suponha que os roteadores R3 e R4 foram conectados um ao outro através de novas interfaces de rede. O que mudará nas tabelas de roteamento?
5. Suponha que o roteador R2 parou de funcionar devido a um problema de hardware. O que poderia ser feito para resolver o problema?

Atividade 2.2 – Problema na tabela de roteamento

Suponha que você é o administrador da rede 10.0.0.0 e, ao investigar um problema com o host 10.0.0.5, descobre que ele está recebendo vários pacotes com diferentes endereços IP de destino e IP de origem 10.0.0.12. Qual seria a causa desse problema?

Atividade 2.3 – Problema na tabela de roteamento

Observe a tabela de rotas a seguir:

Destino	Rota
192.168.0.0/24	interface eth0
192.168.0.7/32	interface PPP0
0.0.0.0	10.0.0.1

Ela está correta? Caso não esteja, o que há de errado com ela? Como resolver o problema?

Atividade 2.4 – Protocolo ICMP

Alguns administradores de rede costumam configurar um firewall de modo a não permitir a saída e a entrada de pacotes do tipo ICMP. Você concorda com isso? O que pode acontecer se isso for feito?

Atividade 2.5 – Comandos ping e traceroute

O comando *traceroute* utiliza pacotes UDP. Seria possível implementar uma espécie de *traceroute* com pacotes TCP?

Atividade 2.6 – Cabeçalho TCP

Sobre o cabeçalho do TCP, responda:

1. O cabeçalho possui tamanho fixo? Quais são as consequências disso?
2. O que você poderia dizer de um pacote TCP com os bits ACK, FIN e SYN ligados?
3. E com os bits SYN e ACK ligados?

Atividade 2.7 – Daemons

Você conseguiria descobrir os daemons que estão rodando em sua máquina? Utilize o comando *netstat*.

Atividade 2.8 – Testando a conexão

Utilize o comando *nc* (netcat) para estabelecer uma conexão com a máquina de outro aluno (dica: nc -l). Você conseguiria descobrir as portas de origem e destino dessa conexão?

Atividade 2.9 – Conexões virtuais

É possível realizar várias conexões para uma mesma porta de destino em um único servidor? Por quê?

Atividade 2.10 – Testando serviços

Como administrador de uma rede, você deve garantir que os serviços oferecidos estejam sempre funcionando. O que você poderia fazer para verificar se o servidor POP3 e o servidor web estão funcionando? Dica: man nc.

3

Configurando uma rede TCP/IP

objetivos

Entender como funcionam os principais equipamentos de rede e conseguir diferenciá-los, configurar interfaces de rede e rotas, entender a resolução de nomes para endereço IP e o modo de fazer atribuições dinâmicas de endereços.

conceitos

Equipamentos de rede, configuração de interfaces de rede e rotas, formas de conexão à internet e configuração de conexão discada, resolução de nomes para endereços IP e atribuição dinâmica de endereços.

Equipamentos de rede

A finalidade dos dispositivos de redes de computadores é servir como mediadores de dados em uma rede – ou simplesmente como equipamentos de rede. Neste capítulo serão vistos os tipos de equipamentos mais comuns encontrados em redes locais.

Hub

O hub, ou concentrador, é um dispositivo de conexão que transmite a mesma informação a múltiplos receptores, atuando como um barramento único que conecta todos os hosts.

O hub funciona como um repetidor com diversas portas, no qual o sinal recebido em uma porta é repetido para as demais. Ao utilizar um hub, cada host conectado compartilha o mesmo domínio de broadcast e de colisão.

Apesar de os hubs serem equipamentos de baixo custo, não devem ser utilizados. Por se tratar de um repetidor em meio compartilhado, um quadro físico é enviado para todas as portas. Com isso, frequentemente ocorrem colisões quando dois ou mais hosts tentam transmitir simultaneamente. Como o mesmo dado é enviado a todas as portas, a utilização de um Hub facilita a captura de pacotes por hosts não autorizados, comprometendo a segurança de uma rede.

Switch

A função de um switch, ou comutador de pacotes, é semelhante à do hub. A diferença é que, no caso do switch, um quadro recebido em uma porta não é repetido em todas as outras portas, mas apenas na porta onde a máquina destino do quadro está conectada. Dessa forma é possível diminuir, ou até mesmo eliminar (para enlaces full duplex), as colisões

quando dois ou mais hosts tentam transmitir ao mesmo tempo. Para tal, o switch armazena na tabela Content Addressable Memory (CAM) o endereço físico dos hosts conectados em cada porta. Após receber um quadro, o switch recupera da tabela o identificador da porta (cujo endereço físico associado é igual ao endereço físico de destino do quadro) e, então, retransmite o quadro apenas na porta identificada. O switch repete o quadro em todas as portas, exceto a porta de entrada do quatro, em duas situações:

- Quando ainda não conhece a porta associada ao endereço físico de destino;
- Quando um quadro é transmitido em broadcast.

Existem switches gerenciáveis, que permitem maior controle por parte dos administradores no que diz respeito ao uso do equipamento. Isso possibilita a divisão do domínio de broadcast em VLANs (virtual LAN), em que um grupo de portas participando de uma VLAN não se comunica diretamente (nível 2, modelo ISO/OSI) com outra VLAN. Além de fazer a função no nível 2, alguns switches também trazem funções de outras camadas.

É importante salientar que um switch analisa apenas os endereços físicos e não os endereços IP.

Roteador

Os roteadores são os equipamentos responsáveis pela interconexão de redes IP, fazendo a passagem de um pacote IP entre as redes. Com vários tipos de interface, pode ligar várias redes IP que funcionam sobre Ethernet, Fiber Distributed Data Interface (FDDI), Asynchronous Transfer Mode (ATM) etc.

Em geral, um roteador não suporta todas as tecnologias de rede simultaneamente, mas apenas um subconjunto. Em adição, diversos roteadores são modulares, permitindo ao administrador selecionar e adquirir apenas os módulos que suportam as tecnologias de rede a serem utilizadas. Caso não esteja disponível um roteador que suporte dois tipos específicos de tecnologia de rede, pode-se utilizar mais de um roteador para conectar essas redes.

Um roteador possui pelo menos duas interfaces de rede, e, em geral, pelo menos um endereço IP por interface, já que cada interface pode estar conectada a uma rede ou a outro roteador. São equipamentos gerenciáveis (o que permite sua administração ou configuração) que utilizam um Sistema Operacional próprio.

Devido ao alto preço dos roteadores, é comum usar equipamentos mais simples para fazer a interconexão de duas ou mais redes. Hoje, o próprio Linux possui suporte no Sistema Operacional para tal função.

Hub

Switch

Roteador

Figura 1.1 Alguns equipamentos de rede bastante comuns.

Configurando endereços e rotas

O dispositivo de rede possui suporte no sistema?

- lsmod
- modprobe <nomedomodulo>
- modinfo

Após o Sistema Operacional obter suporte:

- Comando *ifconfig*.

Configurando endereços IP

Antes de configurar um endereço IP em uma interface de rede, é necessário que o Sistema Operacional saiba "conversar" com o dispositivo (placa de rede) instalado no equipamento. Para isso são utilizados os drivers. Para saber os módulos (drivers) que já foram carregados pelo kernel, utilizamos o comando *lsmod*:

```
# lsmod
Module     Size         Used byNot tainted
pcnet32          29828         1
```

Para carregar um módulo, basta utilizar o comando *insmod* ou o comando *modprobe*.
A diferença entre os dois é que o *modprobe* resolve e carrega as dependências do módulo que estamos tentando carregar. Por exemplo, se o módulo A necessita de que o módulo B esteja carregado, o uso de *modprobe* A carregará ambos.

```
# modprobe pcnet32
```

Alguns módulos aceitam parâmetros ao serem carregados, como, por exemplo, IRQ, I/O, opções de velocidade etc. Para obter mais informações sobre um módulo, utilize o comando *modinfo*. Por exemplo, para obter a lista de parâmetros de um módulo, use *modinfo -p módulo*.

O comando *rmmod* serve para remover um módulo que não está mais em uso. Para obter um efeito análogo ao carregamento de módulos com dependências (para remover, é claro), também podemos usar o comando *modprobe* com a opção *–r.*

Configurando uma interface de rede

Comando *ifconfig:*

- Atribuir um endereço IP:

```
# ifconfig <interface> <endereço IP> netmask <máscara> up
```

- Desabilitar ou habilitar uma interface:

```
# ifconfig <interface> down
# ifconfig <interface> up
```

- Configurando um IP-Aliasing:

```
# ifconfig <interface_real>:<digito> <endereço_IP> netmask
  <máscara>
```

Após o Sistema Operacional obter suporte para o dispositivo de rede utilizado, podemos empregar o comando *ifconfig* para atribuir um endereço IP para esse dispositivo. O *ifconfig* é o principal comando para configuração de interfaces em sistemas Unix ou Linux. A sua sintaxe é:

```
ifconfig [interface]

ifconfig interface [aftype] options | address ...
```

Alguns exemplos são destacados, embora seja aconselhável a leitura do manual (*man ifconfig*).

Exemplos do comando *ifconfig*

Para listar as interfaces e seus endereços:

```
# ifconfig

eth0      Link encap:Ethernet HWaddr 02:60:8C:F1:EB:CF

inet addr:10.10.10.1 Bcast:10.255.255.255 Mask:255.255.255.0

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:11 errors:0 dropped:0 overruns:0 frame:0

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:858 (858.0 b) TX bytes:0 (0.0 b) Interrupt:5 Base
address:0x2440

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

UP LOOPBACK RUNNING MTU:16436 Metric:1

RX packets:55870 errors:0 dropped:0 overruns:0 frame:0

TX packets:55870 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

RX bytes:1245405 (118.7 MiB) TX bytes:1245405 (118.7 MiB)
```

Para atribuir um endereço IP e ativar uma interface:

```
# ifconfig <interface> <endereço IP> netmask <máscara> up
```

Para desabilitar uma interface:

```
# ifconfig <interface> down
```

O Linux possibilita que uma interface tenha vários endereços IP. Isso é chamado de IP-Aliasing, ou seja, uma interface de rede ganha outro IP e passa a agir como se fossem duas placas de rede distintas. Os comandos utilizados para configurar ou manipular interface de rede (como *ifconfig*, *route* e *tcpdump*) podem usar essa interface virtual.

Uma interface virtual (apelido ou alias) é identificada com o formato:

```
interface_real:id.
```

Para criar uma interface virtual e configurar para ela um endereço IP, utilizamos o comando *ifconfig*. Sua sintaxe é:

```
# ifconfig <interface_real>:<id> <endereço_ip> netmask <máscara>
```

Para mudar a MTU de uma interface:

```
# ifconfig <interface> mtu <valor>
```

Para configurar uma interface com uma conexão ponto-a-ponto:

```
# ifconfig <interface> <ender_local> netmask <máscara> pointopoint
<ender_remoto>
```

Para colocar e retirar uma interface do modo promíscuo:

```
# ifconfig <interface> promisc

# ifconfig <interface> -promisc
```

O comando *ifconfig* também pode ser utilizado para alterar o endereço físico da placa. Para isso, é necessário que a placa esteja inativa. Exemplo:

```
# ifconfig <interface> down

# ifconfig <interface> hw ether <endereço_físico>

# ifconfig <interface> up
```

Agora serão destacadas as principais utilizações do comando *ip*. Utilizado com mais frequência desde a versão do kernel 2.4 no Linux, tal comando faz parte do pacote "iproute2" e resume funcionalidades tanto do *ifconfig* quanto do route e do ARP, além de trazer outros comandos para tratar de configurações mais avançadas.

O comando *ip* é utilizado junto a um objeto relacionado a determinadas funções. Sua forma básica é:

```
ip [ OPÇÕES ] OBJETO { COMMAND | help }
```

Mais detalhes sobre o comando ***ip*** podem ser encontrados na sua documentação (geralmente em ***/usr/share/doc/iproute/***).

Objeto	Descrição
link	Configuração de dispositivos de rede.
address	Gerenciamento de endereços IP.
neighbour	Gerenciamento de tabelas ARP.
route	Gerenciamento da tabela de roteamento.
rule	Gerenciamento do banco de dados da política de roteamento.
maddress	Gerenciamento de endereços multicast.
mroute	Gerenciamento de roteamento multicast.
tunnel	Configurações de túneis.

Tabela 3.1
Objetos do comando ***ip*** e suas descrições.

Para obter ajuda simplificada para cada objeto, basta usar a palavra “help” após o nome do objeto:

```
# ip link help
```

Basicamente, todos os objetos possuem a opção *show* (ou *list*) para mostrar mais detalhes. Muitos comandos também podem ser abreviados. No exemplo a seguir, os dois comandos possuem o mesmo significado:

```
# ip l h
# ip link help
```

A seguir veremos alguns exemplos de utilização do comando *ip*, equivalentes aos comandos do *ifconfig* vistos anteriormente. Exemplos do comando *ip* para listar as interfaces e seus endereços:

```
# ip addr show
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
  inet6 ::1/128 scope host
  valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast
qlen 1000
  link/ether 00:0c:29:95:b4:b7 brd ff:ff:ff:ff:ff:ff
  inet 10.211.8.2/16 brd 10.211.255.255 scope global eth0
  inet6 fe80::20c:29ff:fe95:b4b7/64 scope link
  valid_lft forever preferred_lft forever
3: sit0: <NOARP> mtu 1480 qdisc noop
  link/sit 0.0.0.0 brd 0.0.0.0
```

Para ativar ou desativar uma interface:

```
# ip link set <interface> up
# ip link set <interface> down
Para atribuir/remover um endereço IP:
# ip addr add <ip>/<máscara> dev <interface>
# ip addr del <ip>/<máscara> dev <interface>
```

Ao contrário do *ifconfig*, ao atribuir um endereço IP, a interface não fica ativa automaticamente.

O comando *ip* permite a configuração de mais de um endereço IP por interface, sem a necessidade da criação de uma interface virtual. O comando *ip* também pode funcionar como o

ifconfig, configurando endereços IPs adicionais em interfaces virtuais. Um cuidado adicional dever ser tomado quando o comando *ip* for utilizado para adicionar endereços IP sem a criação de interfaces virtuais, já que esses endereços não serão listados pelo comando *ifconfig*.

Para adicionar um endereço IP a uma interface de rede, sem a criação de uma interface virtual:

```
# ip addr add <ip>/<máscara> brd + dev <interface>
```

Para criar uma interface virtual, além de adicionar um endereço IP, acrescentamos a opção "label":

```
# ip addr add <ip>/<máscara> brd + dev <interface> label
<interface>:<id>
```

O exemplo seguinte cria uma interface virtual chamada "eth0:link", com endereço 192.168.0.1/24:

```
# ip addr add 192.168.0.1/24 brd + dev eth0 label eth0:link
```

Para trocar a MTU de uma interface:

```
# ip link set <interface> mtu <valor>
```

Para alterar o endereço físico da placa:

```
# ip link set <interface> down

# ip link set <interface> address <endereço físico>

# ip link set <interface> up
```

Configurando uma interface de rede

Comando *ifconfig*:

- Colocar ou remover modo promíscuo:

```
# ifconfig <interface> prspaomisc

# ifconfig <interface> -promisc
```

- Mudar o valor da MTU:

```
# ifconfig <interface> mtu <valor>
```

Comando *ip*:

```
ip [ OPÇÕES ] OBJETO { COMMAND | help }
```

- Onde:
  - Objeto = { link | addr | route | rule | neigh | tunnel | maddr | mroute | monitor }
  - Opções = { -V[ersion] | -s[tatistics] | -r[esolve] | -f[amily] { inet | inet6 | ipx | dnet | link } | -o[neline] }

Mais ajuda:

```
# ip <objeto> help
```

Configurando uma interface de rede.

Comando *ip*:

- Para listar a interface e seus endereços:

```
# ip addr show
```

- Para habilitar ou desabilitar uma interface:

```
# ip link set <interface> up
# ip link set <interface> down
```

- Para atribuir um endereço IP:

```
# ip addr add <ip>/<máscara> dev <interface>
```

Configurando rotas.

- Pode ser usado o comando *route* ou o comando *ip* .

Listar as rotas:

```
# route
# ip route show
```

Adicionar uma rota default:

```
# route add default gw <gateway>
# ip route add default via <gateway>
```

Rotas para redes:

```
# route add -net <rede> netmask <netmask> gw <gateway>
# ip route add <rede>/<mascara> via <gateway>
```

Para adicionar, remover ou listar as rotas, são usados o comando *route* ou o comando *ip*.

Exemplos dos comandos *route* e *ip*

Para listar as rotas com o comando *route*:

```
# route
Kernel IP routing table
Destination      Gateway Genmask             Flags  Metri  Ref
Useface
10.0.0.0   *      255.255.255.0  U      0      0      0      eth0
Default    10.0.0.1       0.0.0.0 UG     0      0      0      eth0
```

Ou com o comando *ip*:

```
# ip route show
10.0.0.0/24 dev eth0 proto kernel scope link src 10.0.0.1
default via 10.0.0.1 dev eth0
```

Para adicionar a rota default com o comando *route*:

```
# route add default gw <gateway>
```

Ou com o comando *ip*:

```
# ip route add default via <gateway>
```

Para alcançar uma rede, ao adicionar rotas para redes, especifique o endereço IP da rede de destino e do gateway a ser utilizado com o comando *route*:

```
# route add -net <rede> netmask <netmask> gw <gateway>
```

Ou com o comando *ip*:

```
# ip route add <rede>/<bits_mask> via <gateway>
```

Para criar rotas para hosts com o comando *route*:

```
# route add -host <iphost> gw <gateway>
```

Com o comando *ip*, basta utilizar o valor "/32":

```
# ip route add <ip>/32 via <gateway>
```

Configurando endereços e redes

Para automatizar a configuração de rotas, cada distribuição possui locais diferentes.

- No Debian:
  - Arquivo */etc/network/interfaces.*
  - Arquivo */etc/resolv.conf*

Até aqui estudamos a configuração manual de interfaces, endereços e rotas. Obviamente, isso também pode ser feito automaticamente. Porém, cada distribuição Linux coloca seus arquivos de configurações de rede em locais um pouco diferentes. Confira a seguir.

Slackware

Definição de interfaces, endereços e rotas é feita em um único arquivo:

```
/etc/rc.d/rc.inet1
```

Red Hat/Fedora e CentOS

Configuração de interfaces:

```
/etc/sysconfig/network-scripts/ifcfg-*
```

Nome do host e rota padrão (gateway):

```
/etc/sysconfig/network
```

Rotas estáticas:

```
/etc/sysconfig/static-routes
```

Automatizando a configuração de redes e rotas no Debian

Apesar de existirem ferramentas gráficas para a configuração de interfaces de rede e de rotas, é importante que o administrador de redes saiba efetuar essas configurações sem utilizar tais ferramentas. Veremos agora como configurar interfaces de rede e rotas estáticas através de arquivos de configuração.

O primeiro arquivo a ser configurado é o */etc/network/interfaces*. Ele possui, basicamente, dois parâmetros (*auto* e *iface*):

```
auto < interface>

iface < interface> inet <static ou dhcp>
```

O parâmetro *auto* é utilizado apenas para indicar ao sistema que a interface de rede deve ser ativada. O parâmetro *iface* é utilizado para configurar as informações da interface. Por padrão, uma interface pode estar em modo dinâmico (dhcp) ou fixo (static). No primeiro caso todas as configurações IP são recebidas através de um servidor DHCP. Já no segundo caso as configurações de endereçamento são informadas manualmente pelo administrador. Na tabela a seguir vemos os parâmetros que devem ser informados quando for configurada a interface em modo estático.

Parâmetro	Descrição
address	Endereço de rede IP.
netmask	Máscara de rede.
broadcast	Endereço de broadcast.
network	Endereço que identifica a rede.
gateway	Endereço do roteador default da máquina.

Tabela 3.2
Parâmetros de configuração da interface em modo estático.

Um exemplo completo desse arquivo de configuração em modo estático:

```
auto eth0

iface eth0 inet static

address 192.168.0.2

netmask 255.255.255.0

broadcast 192.168.0.255

network 192.168.0.0

gateway 192.168.0.254
```

Interfaces de rede virtuais podem ser configuradas da mesma maneira que as interfaces reais. Basta utilizar o nome da interface virtual (por exemplo, *eth0:0*) no lugar do nome do DEVICE. Por exemplo:

```
auto eth0:0

iface eth0:0 inet static
```

Para configurar as rotas estáticas é necessário adicionar a seguinte linha no arquivo */etc/network/interfaces*:

```
post-up route add -net <rede>/<bits_mask> gw <gateway>
```

As configurações das interfaces podem ser iniciadas, removidas ou reiniciadas através do script de inicialização */etc/init.d/network* seguido das opções *start*, *stop* ou *restart*, respectivamente. Exemplo:

```
# /etc/init.d/networking restart
```

Formas de conexão à internet

Diversas formas de conexão:

- Links dedicados, Frame Relay, ADSL, linha discada (exemplo: 3G) etc.

Protocolo PPP (Point-to-Point Protocol).

Existem muitas formas de se conectar um host ou uma rede à internet. As formas de conexão mais populares, para redes pequenas e redes domésticas, são feitas através de **DSL**, cable modem, rádio (wireless) ou conexões discadas (exemplo: 3G). Para redes maiores, são mais populares outras formas de conexão, como Frame Relay, links seriais dedicados e até mesmo links com fibras óticas.

DSL
Digital Subscriber Line (ou ainda xDSL) é uma família de tecnologias que fornecem um meio de transmissão digital de dados, aproveitando a própria rede de telefonia que chega na maioria das residências.

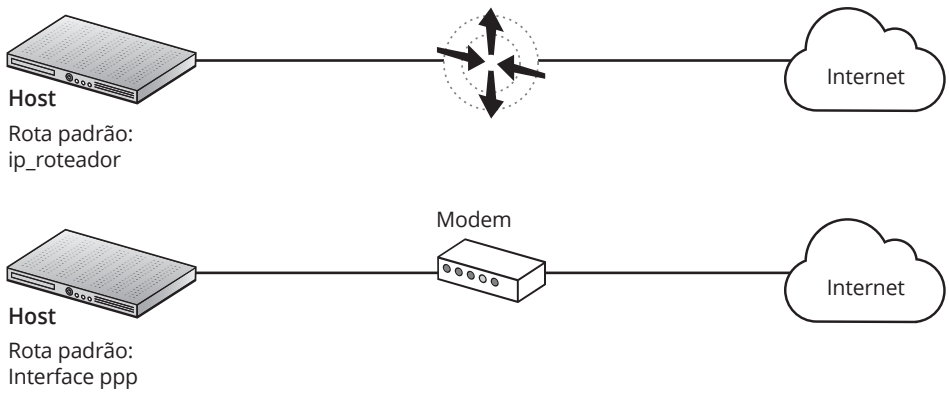


Figura 3.2
Equipamentos necessários para conexão.

Do ponto de vista do administrador de um sistema Unix ou Linux, a forma de conexão pouco determina a maneira como o sistema deve ser configurado, pois basicamente existem dois cenários, conforme ilustrado na figura anterior.

- **Conexão feita via rede Ethernet a um roteador:** a configuração é simples e consiste, basicamente, em se acrescentar a rota default para o roteador (Asymetric Digital Subscriber Line – ADSL);
- **Conexão feita através de um modem**: além da configuração do modem, é necessária a utilização de programas como o PPPD e o chat para fazer a inicialização do modem, a autenticação e o estabelecimento da conexão.

Protocolo PPP

O protocolo ponto a ponto, o PPP e seus protocolos auxiliares são capazes de negociar endereços IP, tamanhos de datagramas e autenticação. Para cada uma dessas capacidades é empregado um protocolo específico como o HDLC (sinalização e correção de quadros), o Link Control Protocol (LCP) e o Password Authentication Protocol (PAP) ou Challenge-Handshake Authentication Protocol (CHAP), estes últimos para autenticação.

No Linux, parte do PPP é implementada no kernel, como os protocolos de nível mais baixo, como o High Level Data Link Control (HDLC). A outra parte é implementada no daemon *pppd*, responsável pela autenticação, entre outras funções.

Resolução de nomes

Biblioteca *resolver* (libresolv). Arquivos:

- /etc/nsswitch.conf
- /etc/host.conf
- /etc/resolv.conf
- /etc/hosts

Com frequência, os diversos programas que utilizam a rede (navegadores e ferramentas como o ping) precisam converter nomes de hosts ou domínios em endereços IP e vice-versa. Nos sistemas Unix ou Linux, essas funções são feitas por meio da biblioteca *resolver* (*libresolv*).

A biblioteca *libresolv* pode utilizar arquivos contendo endereços e nomes de hosts, bases de dados como NIS e servidores de nomes (DNS), que são mais utilizados. Ela é configurada através de alguns arquivos que indicam o modo como a resolução de nomes deve ser feita. Eles são os seguintes:

Arquivo /etc/nsswitch.conf

Possui configurações para uma série de funções da biblioteca C do sistema (libc) e da biblioteca *libresolv*. Esse arquivo lista bases de dados do sistema e serviços. A linha contendo a palavra-chave "hosts" é utilizada para especificar como deve ser feita a resolução de nomes. Em geral, essa linha não precisa ser alterada e seus valores padrão são "files" e "dns". Isso significa que a *libresolv* deve primeiro consultar o arquivo de hosts (/etc/hosts) e, em seguida, caso não encontre o endereço/nome, deve consultar o servidor de DNS. Um trecho do arquivo */etc/nsswitch.conf* é mostrado a seguir:

```
passwd:  files

shadow:  files

group:  files

hosts:  dns files

Arquivo /etc/host.conf
```

Contém algumas configurações específicas da *libresolv*. Em geral, é utilizado para definir a ordem de consulta que a *libresolv* utilizará. O exemplo a seguir do arquivo */etc/hosts.conf* especifica que a *libresolv* deve primeiro consultar o arquivo */etc/hosts* e, em seguida, os servidores de DNS (opção *order*). A configuração a seguir também especifica que a *libresolv* deve retornar todos os endereços do arquivo */etc/hosts* que combinam com a consulta *multi on* feita:

```
order hosts, bind

multi on
```

Em geral, os dois arquivos citados não precisam ser alterados, já que os valores padrão do sistema são adequados.

Arquivo /etc/hosts

Contém uma lista de endereços e nomes de hosts e, como opção, um apelido para o host. Esse arquivo pode ser configurado para resolver nomes de hosts em uma rede local, sem que seja necessária a configuração de um servidor de nomes para esse propósito. Nos primórdios da internet, quando o sistema de DNS ainda não era utilizado, toda resolução de nomes era feita por meio desse arquivo, e ele deveria ser atualizado periodicamente para conter os nomes de hosts da internet da época. Um exemplo de arquivo */etc/hosts*:

```
127.0.0.1          localhost

192.168.0.1        maquina1.meudominio   maquina1

192.168.0.2        maquina2.meudominio   maquina2
```

Arquivo /etc/resolv.conf

Principal opção é o *nameserver.*

- *nameserver <endereço>*

Outras opções:

- Domain.
- Search.

Exemplos:

- *nameserver 60.0.0.1*
- *search rnp.br homelinux.org*

Utilizado para especificar servidores de nomes (DNS). Sua principal palavra-chave é "nameserver" que, seguida de um endereço IP, especifica servidores de nomes. Podem ser configurados até três servidores de nomes, cada qual em uma linha iniciada por "nameserver".

Outras duas opções, *domain* e *search*, permitem o uso de nomes curtos ou nomes de hosts (sem um domínio). Quando fornecido um nome curto, a opção *domain* permite que o domínio informado seja utilizado. A opção *search* é utilizada para especificar uma lista de domínios. Essa lista de domínios será utilizada quando um nome curto for fornecido, adicionando-se os domínios ao nome do host. Por exemplo, ao se tentar resolver o nome de host "teste", se for utilizada a opção "search dominio1.com dominio2.com" a *libresolv* vai procurar por "teste.dominio1.com" e "teste.dominio2.com".

As entradas *domain* e *search* são mutuamente exclusivas, ou seja, apenas uma delas deverá ser utilizada no arquivo */etc/resolv.conf*.

Um exemplo de arquivo */etc/resolv.conf*:

```
search rnp.br homelinux.org

nameserver 60.0.0.1

nameserver 60.0.0.2
```

Ferramentas de consulta a servidores DNS:

- Comando *host.*
- Comando *dig.*

Ferramentas para resolução de nomes

O Linux possui dois comandos que permitem fazer consultas a um servidor DNS para a resolução de nomes e IPs: *dig* e *host*.

O comando *host* aceita, como argumento, um nome ou endereço IP e, como opção, um servidor DNS. Se nenhum servidor for especificado, os servidores listados no arquivo */etc/resolv.conf* serão utilizados. O exemplo a seguir mostra o comando *host* sendo utilizado para resolver o nome "teste.exemplo.com":

```
# host www.google.com.br

www.google.com.br is an alias for www.google.com.

www.google.com is an alias for www.l.google.com.

www.l.google.com has address 72.14.205.104

www.l.google.com has address 72.14.205.147

www.l.google.com has address 72.14.205.99

www.l.google.com has address 72.14.205.103
```

Caso seja fornecido um número IP, o comando *host* retornará o nome ao qual esse IP corresponde (reverso):

```
# host 192.168.0.3

3.0.168.192.in-addr.arpa domain name pointer teste.empresa.com.br.
```

O servidor de nomes deve estar corretamente configurado para que a resolução reversa funcione (de IP para nomes).

O comando *dig* funciona de modo semelhante, ou seja, também pode ser utilizado para efetuar consultas em um servidor DNS para resolver nomes ou IPs. O exemplo a seguir utiliza o comando *dig* para resolver o nome "teste.exemplo.com":

```
# dig www.google.com.br

; <<>> DiG X.X.X <<>> www.google.com.br

;; global options: printcmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48575

;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:

;www.google.com.br.             IN     A

;; ANSWER SECTION:

www.google.com.br.       1690   IN     CNAME  www.google.com.

www.google.com.          82021  IN     CNAME  www.l.google.com.

www.l.google.com. 234    IN     A      72.14.205.99

www.l.google.com. 234    IN     A      72.14.205.103
```

```
www.l.google.com. 234    IN     A      72.14.205.104

www.l.google.com. 234    IN     A      72.14.205.147

;; Query time: 10 msec

;; SERVER: 10.1.2.20#53(10.1.2.20)

;; WHEN: Wed Mar 26 18:40:32 2008

;; MSG SIZE rcvd: 147
```

A saída do comando traz uma série de informações adicionais, como o servidor DNS consultado e detalhes da consulta.

Para se obter o nome do host a partir do IP, deve-se utilizar a opção *-x* do *dig*. Exemplo:

```
# dig -x 192.168.0.3

; <<>> DiG X.X.X <<>> -x 200.238.128.25

;; global options: printcmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52866

;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1,
ADDITIONAL: 1

;; QUESTION SECTION:

;3.0.168.192.in-addr.arpa. IN     PTR

;; ANSWER SECTION:

3.0.168.192.in-addr.arpa  10800  IN     PTR    teste.empresa.com.br.

;; AUTHORITY SECTION:

0.168.192.in-addr.arpa    10800  IN     NS     ns.empresa.com.br.

;; ADDITIONAL SECTION:

ns.empresa.com.br.        10800  IN     A      192.168.0.50

;; Query time: 5 msec

;; SERVER: 192.168.0.50#53(192.168.0.50)

;; WHEN: Wed Jul 28 16:03:11 2004

;; MSG SIZE rcvd: 111
```

DNSSEC

O DNSSEC é um sistema de resolução de nomes que complementa o DNS. Ele é mais seguro, pois garante a origem (autenticidade) e integridade das respostas DNS, reduzindo, assim, o risco de manipulação de dados e informações. Essa funcionalidade é possível por meio da utilização de um mecanismo de chaves públicas, porém o incremento de segurança ainda não cobre questões como confidencialidade e ataques de negação de serviço (DOS).

Todos os domínios ".BR" podem utilizar o DNSSEC, porém o uso dele é obrigatório para as extensões ".B.BR" (bancos) e ".JUS.BR" (entidades do poder judiciário). A possibilidade de utilizar o DNSSEC não invalida o DNS tradicional, sendo assim esta é apenas uma alternativa mais segura.

Para se obter o nome do host a partir do IP, deve-se utilizar a opção *+dnssec* do *dig*. Exemplo:

```
# dig +dnssec +multiline +noadditional www.justica.jus.br

; <<>> DiG X.X.X <<>> +dnssec +multiline +noadditional www.justica.jus.br

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 8787

;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags: do; udp: 4096

;; QUESTION SECTION:

;www.justica.jus.br.      IN A

;; AUTHORITY SECTION:

jus.br.                  180 IN SOA a.dns.br. hostmaster.registro.br.
(

                 2012020908 ; serial

                 1800    ; refresh (30 minutes)

                 900     ; retry (15 minutes)

                 604800   ; expire (1 week)

                 900     ; minimum (15 minutes)

                 )

jus.br.                  180 IN RRSIG SOA 5 2 172800 20120216020000
(

                 20120209020000 24739 jus.br.

                 EOiGzzP9it4GzsM6dLtyICW8KMSJMZqXBeTOtvNkUd/X

                 GH9G6tHfARO+Sd/Rdtj7HLVOoJyQ8+4SKQGg6Re8mrXC

                 9p1lSpY94yuchiN+5PsK2XNE9UhLcVai/S7kRsag7Grz

                 nOOsk5QoCANJkICtPaj+FQlW+ADaYL7zkNKve6o= )

jus.br.                  180 IN RRSIG NSEC 5 2 900 20120214100000 (

                 20120207100000 24739 jus.br.

                 iDWY2n6j7iTwHKiZOwOSjzsDIO+t6mRNPqf6cYUDfQbB

                 lWrUbJUmwlb+FwjhvIjSzbo+JLVf7Pjimu1/QfazPdS1
```

```
                FWuPupIQH+4wJ9zyYh4Pcx3UhIgICzabk5pDDB/MWvwF
                jfi6ntGYMIT12S2la9ldqU1aVyV4wjxOSAkPLMs= )
jus.br.                 180 IN NSEC ac-jus.jus.br. NS SOA RRSIG NSEC
DNSKEY
jttaboaodaserra.jus.br.  180 IN RRSIG NSEC 5 3 900 20120214100000 (
                20120207100000 24739 jus.br.
                UevMhPMDSVL//VanrGLLHHXzOet85aKlvKjBWSbTzMZP
                fAUx33pOf8eOTOLJwHtZ3vUF6Mo1Jx1xAAhmmdfIofaW
                DhDyjFpG9GuUO3D5smOWu2QpIRLHd7AXkGJAI8RbpSU3
                E5DmfTts7AuzFCg1AZWcPSGjcpJ5jc+Kn9UpuOo= )
jttaboaodaserra.jus.br.  180 IN NSEC justicadotrabalho.jus.br. NS DS
RRSIG NSEC
;; Query time: 17 msec
;; SERVER: 10.15.0.10#53(10.15.0.10)
;; WHEN: Thu Feb 9 00:19:52 2012
;; MSG SIZE rcvd: 705
```

Atribuição dinâmica de endereços IP

No Linux existem vários clientes DHCP:

- DHCP Client da ISC.
- Pump.

DHCP Client:

- Arquivo de configuração *dhclient.conf* .
- Via linha de comando:

```
# dhclient <interface>
```

IPv6.

Como foi visto anteriormente, para conectar um host à rede é necessário que este possua um endereço IP, a máscara, a rota default, além de outras informações. Além da forma tradicional já vista, é possível utilizar o protocolo DHCP para essas atribuições.

A principal vantagem de se utilizar o DHCP pode ser vista em redes onde existem equipamentos móveis, tais como notebooks e PDAs, pois a mudança dos hosts da rede é constante. Redes com muitos hosts são ambientes onde o uso de DHCP é interessante, pois permite que o endereçamento seja gerenciado de modo mais centralizado, evitando atribuições duplicadas e outros erros.

Para fazer a atribuição dinâmica de endereços no Sistema Operacional Linux, existem vários clientes DHCP, como o DHCP Client, da Internet Systems Consortium (ISC) (http://www.isc.org) e o *pump* (presente em algumas distribuições de Linux). Aqui, estudaremos a configuração do ISC DHCP Client.

ISC DHCP Client

- Arquivo de posse geralmente em:
  - /var/lib/dhcp3/dhclient.leases
- Arquivo de configuração com diversas opções (dhclient.conf).

O Internet Software Consortium DHCP Client provê um meio para configurar uma ou mais interfaces de rede usando o DHCP, o protocolo BOOTP ou, se estes falharem, atribuir um endereço estático para a interface.

Quando o daemon *dhclient* é iniciado, através da linha de comando ou de algum script de inicialização de interfaces, ele procura no arquivo de configuração *dhclient.conf* por instruções. Assim, pega a lista de todas as interfaces do sistema e configura cada uma delas usando o protocolo DHCP. Também é possível utilizar o *dhclient* via linha de comando:

```
# dhclient eth0
Internet Systems Consortium DHCP Client VX.X.X
Copyright 20XX-20XX Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Listening on LPF/eth0/00:0c:29:95:b4:b7
Sending on  LPF/eth0/00:0c:29:95:b4:b7
Sending on  Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8
DHCPOFFER from 10.211.2.20
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 10.211.2.20
bound to 10.211.8.2 -- renewal in 282134 seconds.
```

Para redes IPv6, deve-se acrescentar o parâmetro ***–6***, desta forma: ***dhclient -6 eth0.***

O nome da interface que se deseja configurar pode ser especificado na linha comando. Porém, se nenhuma interface é informada, o *dhclient* tentará configurar todas elas. Também é possível especificar o nome das interfaces no arquivo de configuração *dhclient.conf*. Se as interfaces não forem informadas na linha de comando, apenas as listadas no arquivo serão configuradas.

Depois de feita a requisição DHCP, as informações de posse do endereço estão geralmente no arquivo */var/lib/dhcp3/dhclient.leases*, que funciona como um banco de dados. As informações disponíveis nesse arquivo podem variar de acordo com a configuração, mas são úteis para o administrador de redes investigar algum problema ou simplesmente checar como foram feitas as atribuições. Exemplo:

```
lease {
interface “eth1”;
fixed-address 192.168.0.33;
```

```
server-name “”;
option subnet-mask 255.255.255.0;
option routers 192.168.0.254;
option dhcp-lease-time 259200;
option dhcp-message-type 5;
option domain-name-servers 192.168.0.254;
option dhcp-server-identifier 192.168.0.254;
option dhcp-renewal-time 129600;
option dhcp-rebinding-time 226800;
option host-name “dhcppc0”;
renew 3 2012/1/17 21:23:00;
rebind 5 2012/1/19 02:27:20;
expire 5 2012/1/19 11:27:20;
}
```

O arquivo de configuração do *dhclient* (/etc/dhcp3/dhclient.conf) possui muitas opções que permitem customizar a requisição feita ao servidor. Listagem de algumas dessas opções:

- **Send**: permite ao cliente enviar informações para o servidor. O objetivo é ajudar o servidor a diferenciar configurações de diversos clientes;
- **Request**: permite que o cliente escolha a quais valores o servidor deverá responder;
- **Require**: lista as opções que devem ser encaminhas pelo servidor para serem aceitas. Se o servidor não retornar todas as opções listadas, a resposta será ignorada;
- **Timeout**: determina o tempo máximo que o cliente deve esperar por uma resposta do servidor. Caso o tempo expire e nenhum servidor responda, pode ser usada a opção *lease* (se estiver definida), ou uma requisição já existente e que ainda não expirou;
- **Retry**: determina o tempo de espera para uma nova tentativa, quando o servidor não é localizado (timeout);
- **Lease**: permite que o administrador atribua as informações que serão utilizadas no host, caso expire o tempo de contatar o servidor (timeout).

Exemplo do arquivo de configuração do DHCP Client, com diversas opções (algumas mencionadas anteriormente):

```
#send host-name “andare.fugue.com”;
#send dhcp-client-identifier 1:0:a0:24:ab:fb:9c;
#send dhcp-lease-time 3600;
#supersede domain-name “fugue.com home.vix.com”;
#prepend domain-name-servers 127.0.0.1;
request subnet-mask, broadcast-address, time-offset, routers,
```

```
   domain-name, domain-name-servers, host-name,
   netbios-name-servers, netbios-scope, interface-mtu;
#require subnet-mask, domain-name-servers;
#timeout 60;
#retry 60;
#reboot 10;
#select-timeout 5;
#initial-interval 2;
#script "/etc/dhcp3/dhclient-script";
#media "-link0 -link1 -link2", "link0 link1";
#reject 192.33.137.209;
#alias {
# interface "eth0";
# fixed-address 192.5.5.213;
# option subnet-mask 255.255.255.255;
#}
#lease {
# interface "eth0";
# fixed-address 192.33.137.200;
# medium "link0 link1";
# option host-name "andare.swiftmedia.com";
# option subnet-mask 255.255.255.0;
# option broadcast-address 192.33.137.255;
# option routers 192.33.137.250;
# option domain-name-servers 127.0.0.1;
# renew 2 2000/1/12 00:00:01;
# rebind 2 2000/1/12 00:00:01;
# expire 2 2000/1/12 00:00:01;
#}
```

Para automatizar a configuração de interface no momento do boot do sistema, informe a utilização de DHCP no arquivo de configuração de interfaces. Nas distribuições Debian, isso é feito com a opção *iface <interface> inet dhcp* no arquivo de configuração da interface. Quando esse parâmetro é utilizado, os scripts de inicialização da interface utilizam o *dhclient* para obter e configurar o endereço IP da interface. O exemplo a seguir, do arquivo */etc/network/interfaces*, utiliza o protocolo DHCP para a configuração da interface *eth0*:

```
auto eth0

iface eth0 inet dhcp
```

Para redes IPv6, substitua o parâmetro *inet* por *inet6* .

IPv6

A internet não foi projetada para ser a grande rede que é hoje. Seu projeto inicial previa somente o uso militar, depois foi aberto aos centros universitários e por fim chegou a ser utilizada comercialmente. Com tamanho crescimento da rede, chegou o esgotamento dos endereços IPs (na versão IPv4).

NAT
Permite que redes privadas se comuniquem com a internet através de um único endereço IPv4.

Medidas paliativas como o Network Address Translation (**NAT**) adiaram o esgotamento dos endereços IPs, mas ainda sim foi necessária a criação da versão 6 desse protocolo.

A principal característica do IPv6 é o incremento de 32 para 128 bits nos endereçamentos, possibilitando capacidade de 79 trilhões de trilhões de vezes a quantidade disponível para versão anterior do protocolo.

Com isso será possível fornecer endereços IPs válidos a todos dispositivos conectados na rede, de um supercomputador a uma geladeira. Além das questões do esgotamento, o novo protocolo proverá mais segurança a rede com a obrigatoriedade do IPSec, reduzirá processamento para os roteadores.

Roteiro de Atividades 3

Atividade 3.1 – Diferença entre switches e hubs

Com base na descrição de switches e hubs, descreva o que aconteceria com um host conectado a cada um desses equipamentos que utilizasse uma interface de rede em modo promíscuo. O que essa interface pode capturar em cada caso?

Atividade 3.2 – Comando *ifconfig*

Para cada item abaixo configure, se necessário, o dispositivo de rede do host e descreva o comando utilizado para fazer a configuração:

1. Após verificar o estado da interface, use o comando *ifconfig*, se necessário, para desativar ou remover o endereço do dispositivo de rede.
2. Com o comando *ifconfig*, ative o dispositivo de rede e atribua o endereço 192.168.0.X/24. Pergunte ao instrutor qual o valor de X.
3. Crie um IP-Aliasing para o dispositivo usando o IP 10.0.0.X/24.

Atividade 3.3 – Comando *ip*

Refaça os itens da Atividade 3.2, agora com o comando *ip*.

Atividade 3.4 – Comando *route*

Reinicie a placa de rede com o comando:

```
# /etc/init.d/networking restart
```

Baseado no comando *route*:

1. Liste a tabela de rotas atual.
2. Remova a rota default, se houver.
3. Adicione uma rota default para a sua rede. Pergunte ao instrutor qual é o gateway da rede.
4. Adicione uma rota para a rede 10.1.0.0/16 com endereço de gateway 192.168.0.X para a sua rede. Pergunte ao instrutor qual o valor de X.

Atividade 3.5 – Comando *ip*

Com o comando *ip*, refaça os itens da Atividade 3.4.

Atividade 3.6 – Rota inválida

Adicione uma rota para outro host da rede (comandos *route* ou *ip*) via um endereço IP não utilizado na rede. Verifique o comportamento da rota usando os comandos *ping* e *traceroute*.

Atividade 3.7 – Configuração de rotas

Com o arquivo de configuração de rede ou rotas:

1. Atribua os valores corretos para deixar a configuração automática.
2. Utilize o script /etc/init.d/networking para parar e iniciar a configuração.
3. Reboot o sistema e verifique se a configuração está funcionando.

Atividade 3.8 – Resolução de nomes

Com o arquivo de resolução de nomes:

1. Localize o arquivo utilizado para resolução de nomes, anote o endereço IP do servidor e o domínio, caso exista.
2. Faça testes de funcionamento do DNS.
3. Verifique o comportamento dos parâmetros *search* ou *domain*.

Atividade 3.9 – DNSSEC

Utilizando o comando *dig*:

1. Verifique a existência de um domínio que utilize o DNSSEC.
2. Verifique quais parâmetros lhe permitem definir se consultas a esse domínio possuem garantia de autenticidade e integridade.

Atividade 3.10 – Atribuição dinâmica de endereços

Com relação à atribuição dinâmica de endereços:

1. Faça uma requisição DHCP com o daemon dhclient, usando o dispositivo apropriado.
2. Analise o arquivo de posse (lease), geralmente em /var/lib/dhcp3/.
3. Altere as configurações do arquivo de configuração /etc/dhcp3/dhclient.conf para requisitar apenas máscara de rede e endereço de broadcast.

4

Verificando a configuração da rede

objetivos

Diagnosticar e solucionar os principais problemas de enlace e roteamento; verificar a configuração das interfaces e entender a correspondência de cada um dos parâmetros apresentados; entender como funciona um sniffer de rede e utilizá-lo para diagnosticar e solucionar problemas da rede; verificar os serviços disponíveis em um host.

conceitos

Enlace e roteamento, configuração de interfaces, correspondência de parâmetros, sniffer de rede e host Linux.st.

Tabela de roteamento

- Comando *route.*
- Comando *ip route.*

Avaliação da rota:

- *traceroute.*
- *ping* (opção de gravar rota).
- *mtr.*

Quando um host consegue acessar normalmente a rede local, mas não consegue acessar outras redes, o problema pode estar em alguma configuração de sua tabela de rotas, como, por exemplo, a ausência de uma rota padrão. Para resolver problemas desse tipo é importante saber como consultar a tabela de rotas da máquina.

Como já vimos, o comando *route* mostra e manipula a tabela de rotamento IP:

```
# route -n
```

Tabela de Roteamento IP do kernel:

```
Destino     Roteador    MáscaraGen.  Opções Métrica Ref  Uso Iface
192.168.0.0     0.0.0.0      255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 eth0
```

O mesmo pode ser feito com o comando *ip*:

```
# ip route list
```

```
192.168.0.0/25 dev eth0 proto kernel scope link src 192.168.0.2

default via 192.168.0.1 dev eth0
```

Para testarmos se uma rota está funcionando, podemos utilizar comandos (ferramentas) como o *ping* e o *traceroute*. Como visto anteriormente, caso ocorra alguma falha, uma mensagem ICMP pode ser retornada:

```
# ping 10.15.0.55

PING 10.15.0.55 (10.15.0.55) 56(84) bytes of data.

From 10.15.0.10 icmp_seq=1 Destination Host Unreachable

From 10.15.0.10 icmp_seq=2 Destination Host Unreachable

From 10.15.0.10 icmp_seq=3 Destination Host Unreachable
```

Ou:

```
# ping 192.168.11.80

ping: sendto: Network is unreachable

ping: wrote 192.168.11.80 64 chars, ret=-1

ping: sendto: Network is unreachable

ping: wrote 192.168.11.80 64 chars, ret=-1
```

Algumas opções do comando *route* também podem ser úteis para avaliarmos as rotas. As flags podem nos indicar as condições mostradas na tabela seguinte.

Flag	Descrição
U	Rota está ativa.
H	Alvo é um host.
G	Uso de gateway.
R	Restabelecer rota para roteamento dinâmico.
D	Dinamicamente instalada por daemon ou ICMP redirect.
M	Modificada pelo daemon de roteamento ou ICMP redirect.
A	Instalada por *addrconf*.
C	Entrada cache.
!	Rota rejeitada.

Tabela 4.1
Flags e descrições.

O Linux possui uma tabela cache de rotas. Nessa tabela são armazenadas as rotas recentemente usadas, em um formato que permite a consulta de maneira mais rápida. Quando um pacote precisa ser roteado, primeiro o kernel consulta essa tabela. Caso a rota não seja encontrada, consulta as demais tabelas de roteamento. Utilizando o comando *route* com a opção -C, a tabela cache será mostrada. Através dela, é possível verificar quais rotas foram recentemente utilizadas pelo sistema. O exemplo a seguir utiliza o comando *route* para mostrar a tabela de roteamento:

```
# route -n -C

Kernel IP routing cache

Source    Destination    GatewayFlags   Metric  Ref    Use    Iface

0.0.0.0    255.255.255.255        255.255.255.255        bl      0
0  4       lo

192.168.0.16       192.168.0.3            192.168.0.3
0  4       3       eth0 192.168.0.16      192.168.0.5
192.168.0.5        0              1       0              eth0

192.168.0.5        192.168.0.16   192.168.0.16   il      0       0
2  lo

192.168.0.3        192.168.0.16   192.168.0.6            il      0
0  7614    lo
```

Avaliação da rota

Existem alguns comandos que servem para avaliar uma rota entre a origem e o destino de um pacote IP. Eles são muito úteis para que se descubra em que ponto da rota pode estar acontecendo um problema.

Comando traceroute

Usado para conhecer por quais roteadores (cada hop) um pacote IP passa até chegar ao seu destino. É possível que os roteadores do caminho possuam filtros ICMP. Por isso, alguns roteadores podem não aparecer na lista.

O único parâmetro obrigatório que o *traceroute* requer é o host de destino. Porém, há outras opções, como -n, que evitam que o traceroute faça a resolução de nomes, restringindo-se a mostrar o endereço IP.

```
# traceroute -n 192.168.63.100

traceroute to (192.168.63.100), 30 hops max, 38 byte packets

1 192.168.255.1 1.024 ms 0.636 ms 0.815 ms

2 192.168.253.106 8.893 ms 9.351 ms 25.867 ms

3 192.168.253.102 29.780 ms 15.169 ms 15.195 ms

4 * * *

5 192.168.255.33 56.366 ms 25.732 ms 17.275 ms

6 192.168.63.190 17.239 ms 17.702 ms 17.741 ms

7 192.168.63.100 22.746 ms 20.178 ms 19.162 ms
```

Para pensar

No exemplo anterior, observe que, por padrão, o *traceroute* aumenta o TTL dos pacotes até o valor máximo de 30 (identificado na saída do comando por 30 hops max). Observe, também, que o quarto hop não respondeu, provavelmente porque possui algum filtro (como um filtro de pacotes UDP ou ICMP, por exemplo), mas que ele está roteando corretamente os pacotes, já que houve resposta do quinto hop.

Comando ping

Ao contrário do *traceroute*, para traçar rotas o *ping* utiliza o campo especial "opções do cabeçalho IP". Quando o valor de rota de registro é utilizado nesse campo, a origem do pacote IP cria uma lista vazia de endereços IP e faz com que seja acrescentado à lista o endereço IP de cada roteador que processe o pacote. Como essa informação é armazenada no cabeçalho do pacote IP, existe um limite de até nove endereços que podem ser armazenados.

Outra característica desse comando é que ele retorna tanto o caminho de ida quanto o de volta do pacote. O exemplo seguinte mostra a utilização do comando *ping* com a opção -R para traçar rotas:

```
# ping -R -n 192.168.25.20

PING (192.168.25.20) 56(124) bytes of data.

64 bytes from 192.168.25.20: icmp_seq=1 ttl=59 time=139 ms

RR:192.168.255.3

192.168.253.105

192.168.253.101

192.168.255.18

192.168.25.161

192.168.25.1

192.168.25.20

192.168.25.162

192.168.255.17

64 bytes from 192.168.25.20: icmp_seq=2 ttl=59 time=126 ms (same route)

64 bytes from 192.168.25.20: icmp_seq=3 ttl=59 time=52.8 ms (same route)

64 bytes from 192.168.25.20: icmp_seq=4 ttl=59 time=247 ms (same route)

--- www.rnp.br ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3035ms rtt
min/avg/max/mdev = 52.889/141.533/247.658/69.575 ms
```

Observe que o caminho de ida começa no host 192.168.255.3 e termina no host de destino 192.168.25.20. O caminho de volta não é mostrado por inteiro, porque o número máximo de endereços armazenados foi atingido.

Comando mtr

Combina a funcionalidade tanto do *ping* quanto do *traceroute*, o que permite um diagnóstico mais preciso. Ao ser iniciado, o *mtr* traça a rota até o endereço IP de destino e, em seguida, envia pings sequencialmente para cada hop. Isso é feito para medir a qualidade do enlace para cada roteador e mostrar tanto a perda quanto o tempo gasto pelos pacotes para percorrer o caminho de ida e volta. O exemplo a seguir mostra a utilização do comando *mtr*:

```
# mtr -n 192.168.25.20

My traceroute [v0.75]

Keys: Help  Display mode  Restart statistics  Order of fields  quit
          Packets               Pings

Hostname          %Loss  Rcv    Snt    Last   Best   Avg    Worst

1. 192.168.255.1    0%     25    25     1      0      1      2

2. 192.168.253.106  0%     25    25     9      8      9      17

3. 192.168.253.102  0%     25    25     15     14     15     15

4. 192.168.255.17   0%     24    25     807    344    800    1667

5. 192.168.25.162   0%     24    25     786    412    772    1078

6. 192.168.25.20    5%     23    25     749    365    759    1057
```

A saída do comando *mtr* mostra:

- Número do host;
- Endereço IP do host;
- Percentual de perda de pacotes;
- Quantidade de pacotes enviados;
- Quantidade de pacotes recebidos;
- Round Trip Time (RTT) do último pacote;
- Menor tempo observado;
- Média de tempo;
- Pior tempo observado.

Verificação do estado do link

- Problema associado com equipamentos.
- Comando *arp* pode ser útil.
- Configurações de velocidade de operação da interface e modo half ou full duplex.
  - Ferramenta *mii-tool.*
    - Hardware pode não ter suporte.

Quando há um host isolado na rede, o primeiro ponto a ser observado é se o problema está relacionado ao link. Alguns exemplos de problemas com o link:

- Cabo mal conectado;
- Porta de switch com defeito;
- Problema na negociação da velocidade do enlace entre um host e um switch.

Os comandos a seguir podem ser úteis na identificação desses problemas.

Comando arp

Pode ajudar na identificação e localização de um possível problema com o cabeamento ou equipamento de rede. Se a entrada na tabela ARP para uma determinada máquina está vazia, provavelmente esta máquina tem um problema na configuração da interface ou no cabeamento físico. Se as tabelas ARP de todas as máquinas estão vazias, provavelmente existe um problema no hub ou switch da rede.

Alguns problemas podem ocorrer quando se negocia a velocidade da interface de um host com um switch. Nesse caso, o comando *mii-tool* pode servir para verificar e alterar o estado Media-Independent Interface (MII) de uma interface.

Para verificar o estado de uma interface, basta usar o comando sem parâmetros. Exemplo:

```
# mii-tool

eth0: negotiated 100baseTx-FD, link ok
```

Para forçar que uma placa anuncie somente certas capacidades ou para desabilitar completamente a autonegociação e forçar uma certa velocidade, use os parâmetros -A e -F, respectivamente. Exemplos:

Anuncia que a placa entende 100 Mbit/s half e full duplex:

```
# mii-tool -A 100BaseTx-FD,100BaseTx-HD eth0
```

Força a placa a entrar no modo 10 Mbit/s full duplex:

```
# mii-tool -F 10BaseT-FD eth0
```

Verificação da configuração das interfaces

Comando *ifconfig*:

Observar todas as informações:

- Erros ou pacotes descartados.
- Flags (UP, PROMISC etc.).

Os comandos *ifconfig* e *ip* mostram as configurações da interface. Algumas informações sobre a interface podem ser observadas ao se utilizar o comando *ifconfig*:

```
# ifconfig

eth0    Encapsulamento do Link: Ethernet Endereço de HW
00:0C:29:95:B4:B7

     inet end.: 10.211.8.2 Bcast:10.211.255.255 Masc:255.255.0.0

     endereço inet6: fe80::20c:29ff:fe95:b4b7/64 Escopo:Link

     UP BROADCASTRUNNING MULTICAST MTU:1500 Métrica:1

     RX packets:1021 errors:0 dropped:0 overruns:0 frame:0

     TX packets:890 errors:0 dropped:0 overruns:0 carrier:0

     colisões:0 txqueuelen:1000

     RX bytes:98100 (95.8 KiB) TX bytes:96686 (94.4 KiB)

     IRQ:185 Endereço de E/S:0x1080

lo     Encapsulamento do Link: Loopback Local

     inet end.: 127.0.0.1 Masc:255.0.0.0

     endereço inet6: ::1/128 Escopo:Máquina

     UP LOOPBACKRUNNING MTU:16436 Métrica:1

     RX packets:66 errors:0 dropped:0 overruns:0 frame:0

     TX packets:66 errors:0 dropped:0 overruns:0 carrier:0

     colisões:0 txqueuelen:0

     RX bytes:5104 (4.9 KiB) TX bytes:5104 (4.9 KiB)
```

As estatísticas de utilização também podem ser úteis ao administrador, para avaliar o uso do dispositivo e como esse uso afeta o resto da rede. Taxas de erros altas, por exemplo, podem indicar algum problema com o adaptador ou com o cabeamento.

Os parâmetros *errors* e *dropped* devem ser observados, pois podem indicar algum problema no cabeamento, switch/hub ou configuração de velocidade da interface. As flags também indicam o estado do dispositivo. Lista de flags:

- **UP**: dispositivo está funcionando;
- **BROADCAST**: dispositivo pode enviar tráfego para todos os hosts localizados na mesma rede;
- **RUNNING**: dispositivo está operacional;
- **MULTICAST**: dispositivo pode atuar e receber pacotes multicast;
- **ALLMULTI**: dispositivo pode receber todos os pacotes multicast presentes na rede em que ele se encontra;
- **PROMISC**: dispositivo pode receber pacotes com qualquer endereço MAC.

Sniffers

BSD Packet Filter (BPF):

- Facilidade para capturar pacotes em espaço de usuário, permitindo o uso de estações de trabalho para monitoramento da rede de forma eficiente.

libpcap:

- Interface independente de sistema para capturar pacotes em user-level que suporta mecanismos de filtragem baseados em BPF.
- tcpdump e Ethereal são os principais sniffers que a suportam.

Sniffers são softwares que capturam o tráfego de rede, extremamente úteis para que o administrador possa depurar problemas e analisar o tráfego.

Quando um quadro é enviado por um determinado host em uma rede **Ethernet**, ele pode ser capturado por outros hosts, mesmo que esses hosts não sejam o destino desse quadro. Basta configurar a interface de rede no modo promíscuo, que ela processará não apenas os quadros enviados para o endereço físico da interface, mas também qualquer outro quadro no barramento. Para que seja possível capturar pacotes, eles devem atravessar o mesmo meio físico em que o host está conectado. Em geral, a forma mais simples de se fazer isso é através da utilização de um hub.

Ethernet
Protocolo de interconexão para redes locais – Rede de Área Local (LAN) – baseado no envio de pacotes.

Antes de falarmos dos principais sniffers, iremos verificar uma funcionalidade disponível no kernel dos principais Sistemas Operacionais: o Berkeley Packet Filter (BPF).

BPF

O BPF é uma facilidade para capturar pacotes em espaço de usuário (user-level), permitindo o uso de estações de trabalho para monitoramento da rede de forma eficiente. Como monitores de rede rodam no espaço de processos user-level, os pacotes devem ser copiados para um limite permitido, para que possam ser manipulados pelos monitores. Essa cópia é feita por um agente do kernel chamado packet filter. O BPF permite que essa cópia seja feita de forma muito eficiente, já que não funciona sobre a pilha de protocolos TCP/IP como os antigos packet filters, mas sim baseado num novo registro e com um bom sistema de área de armazenamento (buffer) que permite que seja muitas vezes mais rápido.

libpcap

Interface independente de sistema para capturar pacotes em user-level, que suporta mecanismos de filtragem baseados em BPF. A seguir, veremos os principais sniffers que suportam a libpcap, como o tcpdump e o Wireshark.

tcpdump

Imprimir todos os pacotes na interface *eth1* sem resolução de nomes de hosts e portas:

```
# tcpdump -n -i eth1
```

Imprimir todo o tráfego entre a rede local e a rede 192.168.0.0/24

```
# tcpdump dst net 192.168.0.0/24
```

Poderosa ferramenta de monitoramento e sniffer de rede, que suporta muitos protocolos como Ipv4; ICMPv4; IPv6; ICMPv6; UDP; TCP; BGP; OSPF; SNMP; AFS; RIP; PIM; DVMRP; IGMP; SMB; NFS; entre outros tipos. O tcpdump imprime o cabeçalho de pacotes que casam com uma expressão (filtros BPF) booleana. Exemplos de utilização do tcpdump.

Para imprimir todos os pacotes da interface *eth1*, sem resolução de nomes de hosts e portas:

```
# tcpdump -n -i eth1
```

Para imprimir tráfego do host diablo:

```
# tcpdump host diablo
```

Para imprimir todos os pacotes IPs entre o host maverick e qualquer host, exceto a galaxie:

```
# tcpdump ip host maverick and not galaxie
```

Para imprimir todo o tráfego entre a rede local e a rede 10.0.0.0/24:

```
# tcpdump dst net 10.0.0.0/24
```

Para imprimir todos os pacotes ICMP que não são echo request ou reply:

```
# tcpdump 'icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-
echoreply'
```

Ethereal (Wireshark)

O Wireshark antigo (Ethereal) é um analisador de tráfego de rede (sniffer), que possui uma interface GUI e captura pacotes no formato libpcap. O Ethereal mostra três janelas de visualização de um pacote:

- Uma janela contendo uma lista de pacotes capturados (1);
- Uma janela com detalhes de um pacote selecionado (2);
- Uma janela que mostra o conteúdo do pacote selecionado no formato hexadecimal e ASCII (3).

A figura a seguir mostra a interface do Ethereal para configurar a captura de pacotes. Filtros BPF podem ser criados e salvos para uso futuro.

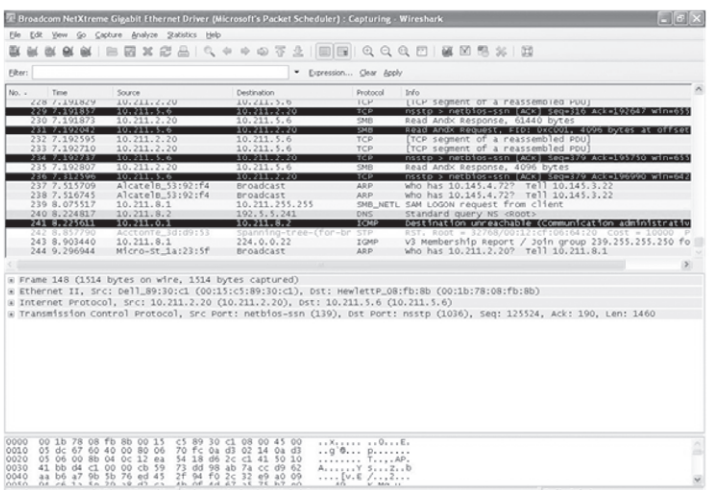


Figura 4.1
Wireshark: interface com três janelas.

Verificação de serviços

Processo associado ao daemon.

Sockets:

- Utilizados na implementação de um serviço para que um processo tenha acesso à pilha de protocolos de rede que são implementados no kernel do Sistema Operacional.

Alguns comandos são úteis para a verificação de serviços, como, por exemplo, o *netstat*, o *lsof* e o *nmap*. Antes de verificar o estado de um serviço, é necessário verificar se o processo que oferece o serviço está ou não sendo executado. Para isso, basta utilizar o comando *ps* e procurar pelo processo em questão.

O exemplo a seguir utiliza o comando *ps* para listar processos e o comando *grep* para selecionar as linhas que contêm a string *named*, que é nome de um daemon.

```
# ps xuaw | grep named
root   2336 0.0 1.3  2864 1320 ?    Ss  12:42  0:00 /usr/sbin/named
root   2981 0.0 0.7  3096  672 pts/0  R+  13:01  0:00 grep named
```

A interface de socket do Linux é baseada na interface criada pelo sistema BSD, e, por isso mesmo, também é conhecida como BSD socket.

Para mais detalhes, consulte a página de manual do comando ps (man ps).

As opções *xuaw* do comando *ps* especificam que todos os processos de todos os usuários devem ser listados no formato *wide*.

Sockets

TCP:

- Família:
  - PF_INET ou PF_INET6.
- Socket:
  - SOCK_STREAM.

UDP:

- Família:
  - PF_INET ou PF_INET6.
- Socket:
  - SOCK_DGRAM.

Para saber mais detalhes dos serviços, é necessário ter noção de como eles são implementados em sistemas Unix ou Linux.

Os protocolos de rede como o IP, o TCP e o UDP são implementados diretamente no kernel do Linux. Para que uma aplicação possa se comunicar utilizando esses protocolos, é necessário utilizar uma Aplication Programing Interface (API), que no Linux é chamada de socket.

Nos sistemas Unix ou Linux, quando uma aplicação precisa se comunicar com outra (na mesma máquina ou em uma máquina remota), ela utiliza um socket. Nos sistemas Unix ou Linux, um socket é tratado de maneira bastante similar a um arquivo, aceitando chamadas de sistema como *read* e *write* (ler e escrever), abstraindo para o programador detalhes dos protocolos de comunicação. Sockets entendem diversas famílias de protocolos e podem ser divididos em vários tipos:

Família	Descrição
PF_INET	Protocolo Internet IPv4.
PF_INET6	Protocolos Internet IPv6.
PF_IPX	Protocolos Novell.
PF_PACKET	Protocolos no nível 2.
Socket	**Descrição**
SOCK_STREAM	Utiliza o protocolo TCP.
SOCK_DGRAM	Utiliza o protocolo UDP.

Tabela 4.2
Sockets.

Para verificar um serviço, é importante ter noção de como ele foi implementado. Ou seja, se foram utilizados a família PF_INET e o socket do tipo SOCK_STREAM, é possível deduzir que esse é um serviço sobre IPv4 que utiliza TCP. Se o serviço foi implementado utilizando PF_INET6 e SOCK_DGRAM, sabemos que é um serviço sobre IPv6 que usa o protocolo de transporte UDP.

Comando netstat

```
# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address  Foreign Address  State
tcp       0      0   0.0.0.0:110     0.0.0.0:*         LISTEN
tcp       0      0   192.168.0.2:53  0.0.0.0:*         LISTEN
tcp       0      0   127.0.0.1:53    0.0.0.0:*         LISTEN
```

O comando *netstat* fornece várias informações relacionadas com a rede, como, por exemplo, quantas portas estão aguardando conexões; qual o protocolo de transporte utilizado; qual o status das conexões etc.

```
# netstat -an
Conexões Internet Ativas (servidores e estabelecidas)
Proto Recv-Q Send-Q Endereço Local  Endereço Remoto  Estado
tcp   0      0      0.0.0.0:110      0.0.0.0:*         LISTEN
tcp   0      0      192.168.02:53    0.0.0.0:*         LISTEN
tcp   0      0      127.0.0.1:53     0.0.0.0:*         LISTEN
tcp   0      0      0.0.0.0:22       0.0.0.0:*         LISTEN
tcp   0      0      0.0.0.0:25       0.0.0.0:*         LISTEN
tcp   0      0      192.168.02:22    192.168.05:40162  ESTABLISHED
tcp   0      0      192.168.02:110   10.0.0.3:39297    ESTABLISHED
udp   0      0      192.168.02:53    0.0.0.0:*
udp   0      0      127.0.0.1:53     0.0.0.0:*
udp   0      0      0.0.0.0:3797     0.0.0.0:*
```

No exemplo anterior, é possível identificar vários serviços aguardando conexões (estado LISTEN): POP3 (porta 110); DNS (porta 53); SSH (porta 22); SMTP (porta 25). Também é possível verificar que existem duas conexões estabelecidas (ESTABLISHED); um SSH; outra POP3; e que alguns serviços também estão abrindo portas usando o protocolo de transporte UDP.

Estado	Descrição
ESTABLISHED	O socket tem uma conexão estabelecida.
SYN_SENT	O socket está tentando estabelecer uma conexão.
SYN_RECV	A requisição de conexão foi recebida da rede.
FIN_WAIT1	O socket está fechado e a conexão está sendo finalizada.
FIN_WAIT2	A conexão está fechada e o socket está aguardando finalizar o lado remoto.
TIME_WAIT	O socket está esperando para tratar pacotes que ainda estão na rede após o socket ter fechado.
CLOSE	O socket não está sendo usado.
CLOSE_WAIT	O lado remoto finalizou, aguardando o socket fechar.
LAST_ACK	O lado remoto finalizou e fechou o socket, aguardando apenas o reconhecimento (fin-ack).
LISTEN	O socket está escutando por conexões.
CLOSING	Ambos os sockets estão finalizando, mas ainda não se tem todos os dados enviados.
UNKNOW	O estado do socket não é conhecido.

Tabela 4.3
Comando *netstat*.

A tabela anterior mostra uma lista de estados possíveis para uma conexão.

Argumento	Descrição
-l	Listas apenas sockets em estado LISTEN (observe que os sockets para o protocolo UDP, apesar de não possuírem estado LISTEN, também são listados).
-n	Mostra em forma numérica, ao invés de tentar resolver nomes e portas.
-s	Mostra uma sumarização para cada protocolo.
-c	A cada segundo a saída será continuamente atualizada.
-p	Será mostrado o PID e o nome do programa ao qual o socket pertence.
-r	Mostra a tabela de rotas do kernel.

Tabela 4.4
Argumentos comuns.

A utilização do comando *netstat* é bastante ampla.

Comando lsof

Um socket é tratado pelo SO como um arquivo:

```
# lsof -n -P -i
COMMAND   PID    USER   FD     TYPE   DEVICE SIZE   NODE
                NAME
ssh             1569   ruth   3u     IPv4                5421
  TCP 192.168.0.5:33508->192.168.0.6:22 (ESTABLISHED)
mozilla   2662   ruth   17u    IPv4                72753  TCP
  192.168.0.5:40852->192.168.0.6:80     (CLOSE_WAIT)
ssh             26746  ruth   3u     IPv4                72446
  TCP 192.168.0.5:40782->192.168.0.2:22 (ESTABLISHED)
ssh             26830  ruth   3u     IPv4                72582
  TCP 192.168.0.5:40809->192.168.0.2:22 (ESTABLISHED)
```

Outro comando que pode ser útil para verificar o estado de um serviço é o *lsof*, que lista todos os arquivos abertos em diversos sistemas Unix ou Linux. Visto que, em um sistema Unix ou Linux, um socket é tratado como um arquivo, ele pode ser listado. A seguir veremos algumas formas de utilização do comando *lsof*, que podem ser úteis para a monitoração de serviços:

- **-n** – inibe a conversão de endereços de redes para nome de hosts, permitindo uma execução mais rápida.
- **-P** – inibe a conversão de números de portas para nomes de portas.
- **-i [i]** – seleciona os arquivos que pertencem ao endereçamento internet, que casam com o padrão especificado em "i". Onde "i" pode ser especificado da seguinte forma: *[protocolo][@nomehost|endereç ohost][:serviç o|porta]*

Isso significa que é possível selecionar arquivos com eficácia, usando o protocolo (TCP ou UDP); o nome ou endereço do host; o nome do serviço (*/etc/services*) ou a porta.

Para listar todos os sockets que pertencem ao endereçamento internet sem qualquer tradução de nomes:

```
# lsof -n -P -i
COMMAND    PID      USER  FD  TYPE  DEVICE SIZE/OFF NODE NAME
portmap  1376     daemon  4u IPv4   4680   0t0 UDP *:111
portmap  1376     daemon  5u IPv4   4691   0t0 TCP *:111 (LISTEN)
rpc.statd 1392     statd  4u IPv4   4748   0t0 UDP *:720
rpc.statd 1392     statd  6u IPv4   4758   0t0 UDP *:39841
rpc.statd 1392     statd  7u IPv4   4761   0t0 TCP *:42243 (LISTEN)
zabbix_ag 1735     zabbix  5u IPv4   5642   0t0 TCP *:10050 (LISTEN)
zabbix_ag 1754     zabbix  5u IPv4   5642   0t0 TCP *:10050 (LISTEN)
zabbix_ag 1755     zabbix  5u IPv4   5642   0t0 TCP *:10050 (LISTEN)
```

Para listar os arquivos associados a uma conexão TCP na porta 22:

```
# lsof -i tcp:22 -n -P

COMMAND PID USER  FD  TYPE  DEVICE SIZE/OFF NODE NAME

sshd  1873 root  3u IPv4   5937   0t0 TCP *:22 (LISTEN)

sshd  1873 root  4u IPv6   5939   0t0 TCP *:22 (LISTEN)

sshd  9810 root  3r IPv4 265866299   0t0 TCP 10.15.0.10:22-
>10.15.104.151:43170 (ESTABLISHED)

sshd  9868 root  3r IPv4 265866690   0t0 TCP 10.15.0.10:22-
>10.15.105.226:49255 (ESTABLISHED)
```

Para listar os sockets que utilizam o protocolo UDP no serviço DNS (porta 53) para o localhost:

```
# lsof -i udp@localhost:domain -n -P

COMMAND  PID USER  FD  TYPE  DEVICE SIZE/OFF NODE NAME

named  31642 bind 513u IPv4 166390003   0t0 UDP 127.0.0.1:53

named  31643 bind 513u IPv4 166390003   0t0 UDP 127.0.0.1:53

named  31644 bind 513u IPv4 166390003   0t0 UDP 127.0.0.1:53
```

Comando nmap

- Scanners (varredores).
- Usar com cuidado.
- Pode ser utilizado para identificar portas ou estados.
  - Open.
  - Filtered.
  - Unfiltered ou Closed.

Ferramenta de exploração de serviços na rede, que pode ser utilizada para monitorar serviços. As ferramentas que fazem exploração de serviços, checando os serviços no hosts da rede, são conhecidas como scanners (varredores) de rede.

Essas ferramentas devem ser utilizadas com cuidado, pois podem ferir as políticas de segurança de alguma instituição. Isso porque elas podem ser usadas por alguém mal-intencionado para checar os serviços ativos, obtendo assim informações que podem ser utilizadas para alguma ação maliciosa.

Para conhecer outras combinações de utilização do comando *lsof*, verifique o manual (*man lsof*).

O resultado de um *nmap* retorna uma lista de portas e o seu estado, que pode ser:

- **Open**: significa que a porta varrida aceitará conexões;
- **Filtered**: significa que um roteador ou firewall não permitiu que o *nmap* determinasse o estado de *open* na porta;
- **Unfiltered ou closed**: significa que a porta parece estar fechada e nenhum conjunto de regras ou firewall no caminho está interferindo.

```
# nmap 10.15.0.10 -p20-30
Starting Nmap 5.00 ( http://nmap.org ) at 2012-03-07 16:30 BRT
Interesting ports on 10.15.0.10:
PORT  STATE SERVICE
20/tcp closed ftp-data
21/tcp open  ftp
22/tcp open  ssh
23/tcp closed telnet
24/tcp closed priv-mail
25/tcp closed smtp
26/tcp closed rsftp
27/tcp closed nsw-fe
28/tcp closed unknown
29/tcp closed msg-icp
30/tcp closed unknown
Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
```

Para varrer um host, basta informar o endereço IP. Por padrão, o *nmap* não varrerá todas as 65.536 portas, mas apenas aquelas com serviços conhecidos no seu arquivo de serviços (geralmente em */usr/share/nmap/nmap-services*).

```
# nmap 127.0.0.1
Starting Nmap 5.00 ( http://nmap.org ) at 2012-03-07 16:31 BRT
Interesting ports on localhost (127.0.0.1):
Not shown: 988 closed ports
PORT   STATE SERVICE
21/tcp  open ftp
22/tcp  open ssh
25/tcp  open smtp
53/tcp  open domain
80/tcp  open http
111/tcp open rpcbind
139/tcp open netbios-ssn
389/tcp open ldap
445/tcp open microsoft-ds
631/tcp open ipp
```

```
3306/tcp open mysql

5666/tcp open nrpe

Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
```

O exemplo seguinte pode ser utilizado para localizar os hosts dentro da rede 10.15.0.0/27 que aceitam conexões na porta 22 (SSH):

```
# nmap 10.15.0.0/27 -p 22

Starting Nmap 5.00 ( http://nmap.org ) at 2012-03-07 16:32 BRT

Interesting ports on 10.15.0.9:

PORT  STATE SERVICE

22/tcp open ssh

MAC Address: 00:22:19:55:AB:27 (Dell)

Interesting ports on 10.15.0.21:

PORT  STATE SERVICE

22/tcp open ssh

MAC Address: 00:40:F4:F0:1F:36 (Cameo Communications)

Nmap done: 32 IP addresses (9 hosts up) scanned in 1.58 seconds
```

Para varrer as portas entre 20 e 30, o intervalo pode ser informado da seguinte forma:

```
# nmap 10.15.50.13 -p 20-30

Starting Nmap 5.00 ( http://nmap.org ) at 2012-03-07 16:33 BRT

Interesting ports on 10.15.50.13:

PORT  STATE SERVICE

20/tcp closed ftp-data

21/tcp open  ftp

22/tcp open  ssh

23/tcp closed telnet

24/tcp closed priv-mail

25/tcp closed smtp

26/tcp closed rsftp

27/tcp closed nsw-fe

28/tcp closed unknown

29/tcp closed msg-icp

30/tcp closed unknown

MAC Address: 00:0C:29:86:A5:F3 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.43 seconds
```

Roteiro de Atividades 4

Atividade 4.1 – Tabela de Roteamento

Instalar o pacote *mtr*:

```
# apt-get install mtr
```

A partir dos conceitos estudados para avaliar a tabela de roteamento:

1. Descreva quais são as informações relevantes ao se consultar a tabela de roteamento. Use o comando *route*.
2. Adicione uma rota para um host local e descreva o que aconteceu com a tabela de roteamento. Verifique as flags.
3. Utilize o comando *ping* para enviar pacotes ICMP para o host que foi adicionado à rota e com a opção -C do comando *route*. Verifique a utilização da rota.
4. Utilize as ferramentas *route, ping -R* e *mtr* para avaliar a rota apresentada e explique a saída de cada uma. Como destino, utilize um endereço fornecido pelo instrutor.

Atividade 4.2 – Comando *ifconfig* e/ou *ip link*

Com os comandos *ifconfig* e *ip link*:

1. Descreva as informações observadas.
2. Usando apenas o comando *ifconfig*, configure a interface para modo promíscuo. Quais flags foram alteradas (verifique com os comandos *ip* e *ifconfig*)? O que isso significa?
3. Retire a interface do modo promíscuo.

Atividade 4.3 – Comando tcpdump

Instalar o pacote *tcpdump*:

```
# apt-get install tcpdump
```

Com o sniffer *tcpdump*:

1. Com o tráfego observado, é possível concluir se existe na rede um switch ou um hub?
2. Utilize a opção do *tcpdump* para não resolver nomes. É possível notar alguma diferença de desempenho? Justifique.
3. Quais protocolos podem ser identificados?
4. Escolha um host na rede e monitore seu tráfego com os parâmetros apropriados no *tcpdump*.
5. Faça um *ping* para um host na rede. Use o *tcpdump* com os parâmetros apropriados para monitorar apenas os pacotes relacionados com o *ping*.

Atividade 4.4 – Wireshark (Ethereal)

Instalar o pacote Ethereal:

```
# apt-get install ethereal
```

Com o Wireshark, repita os passos da Atividade 4.3. Que vantagens podem ser observadas ao utilizar o Wireshark?

Atividade 4.5 – Comandos *netstat* e/ou *lsof*

Use os comandos *netstat* e/ou *lsof* para monitoramento de serviços:

1. Identifique os serviços que estão no estado LISTEN.
2. Use o comando *nc* como servidor (nc -l -p <porta>) para escutar em uma porta qualquer.
3. Verifique novamente os serviços em estado LISTEN.
4. Use o comando *nc* como cliente (nc <ip> <porta>) para conectar ao servidor e verifique o estado das conexões continuadamente. O que pode ser observado?

Atividade 4.6 – Serviços

Faça testes rodando dois ou mais serviços sobre a mesma porta. Primeiro, use somente o TCP; em seguida, somente o UDP; por fim, TCP e UDP. Justifique o comportamento.

Atividade 4.7 – Comando nmap

Instalar o pacote *nmap*:

```
# apt-get install nmap

Com o comando nmap, responda:
```

1. Quais são as portas abertas no endereço 127.0.0.1?
2. Procure os hosts na sua rede que estão com a porta 22 aberta.
3. Localize nos hosts de IP, com o último octeto variando de 1 a 100 (X.X.X.1-100), os serviços que rodam entre as portas 20 e 50.

Atividade 4.8 – Identificando serviços com o nmap

Use o comando *nc* como servidor (*nc -l -p <porta>*) em portas com serviços conhecidos (TCP e UDP). Solicite que outro aluno em outro host descubra e classifique as portas e serviços utilizando o *nmap*.

Atividade 4.9 – Monitoramento de tráfego de rede

Para resolver esta atividade, aguarde o instrutor iniciar o exercício. O instrutor coordenará a divisão da turma em duas equipes. Durante um período determinado pelo instrutor, as equipes deverão atuar como geradores de tráfego e, depois, como monitores de tráfego.

Durante o período de geração de tráfego, utilize os comandos *ping*, *nc* e *nmap* para gerar tráfego com destino às máquinas dos outros alunos. Durante o período de monitoração de tráfego, utilize o sniffer Wireshark para capturar pacotes.

Ao final da atividade, utilize o Wireshark para analisar o tráfego de rede capturado. Liste a maior quantidade de tráfego que conseguir, bem como a maior quantidade possível de detalhes, como hosts de origem do tráfego capturado, tipo de tráfego gerado etc. Exemplo: "host x.x.x.x efetuou requisições ICMP (pings); host x.x.x.y tentou uma conexão TCP na porta ZZ".

5

Segurança – Introdução

objetivos

Gerenciar e controlar o acesso aos serviços, configurar e utilizar o shell seguro (SSH), entender como os arquivos de log podem ser úteis na detecção de uma invasão, saber como agir em caso de incidentes de segurança e conhecer medidas preventivas de segurança.

conceitos

Atividades associadas à administração de redes.

Segurança

Envolve o conhecimento de vários assuntos.

- Administração de sistemas.
- Sistemas Operacionais.
- Sistemas de arquivos.
- Protocolos de rede.

Para ter um ambiente seguro é necessário observar, no mínimo, cinco princípios:

- Confidencialidade.
- Integridade.
- Disponibilidade.
- Não repúdio (irretratabilidade).
- Autenticidade.

A segurança da informação pode ser definida como a área de conhecimento dedicada à proteção de ativos contra acessos não autorizados e alterações indevidas. De forma geral, um sistema computacional sempre estará vulnerável e poderá ser comprometido, seja por uma falha recém-descoberta ou por "desleixo" de um administrador descuidado.

Para se manter um ambiente computacional seguro é necessário que o administrador possua conhecimentos sobre o Sistema Operacional, protocolos de rede e sistemas de arquivos dos servidores e equipamentos utilizados. De forma geral, o objetivo de um administrador de rede é garantir para todo o ambiente:

- **Confidencialidade**: proteger as informações contra o acesso de pessoas não explicitamente autorizadas pelo dono da informação. Computacionalmente, para se garantir esse princípio pode-se fazer uso de criptografia ou de regras simples de acesso;
- **Integridade**: evitar que os dados sejam apagados ou de alguma forma alterados, sem a permissão do proprietário da informação. Para cumprir esse objetivo, pode-se fazer uso de algoritmos de hash que, ao processarem uma determinada informação, fornecem um número único que identifica aquela porção de bits. Qualquer alteração nos bits originais vai gerar um número totalmente diferente permitindo, assim, identificar claramente quando a informação foi alterada;
- **Disponibilidade**: proteger os recursos de forma que não fiquem indisponíveis sem a devida autorização. Em termos gerais, esse princípio pode incorporar a duplicação do ativo ou o uso de equipamentos de tolerância a falhas;
- **Autenticação**: capacidade de garantir que um usuário, sistema ou informação seja mesmo quem alega ser. Esse princípio pode ser garantido utilizando os três pilares de identificação: o que eu sei (exemplo: senhas), o que eu tenho (exemplo: cartão, token etc.) e o que eu sou (exemplo: biometria). Um sistema é definido como forte se combinar dois ou mais desses fatores no seu processo de autenticação;
- **Não repúdio ou irretratabilidade**: capacidade do sistema de provar que um usuário executou determinada ação no sistema. Esse talvez seja o princípio mais complicado de implementar no mundo computacional. Como garantir que nenhuma das partes poderá negar a realização de uma determinada transação, já que os usuários envolvidos podem estar a quilômetros de distância um do outro? A resposta a esse questionamento vem da aplicação da criptografia assimétrica filiada ao uso de certificados digitais;
- **Legalidade**: aderência de um sistema à legislação. O sistema deve estar em conformidade com a política de segurança interna da organização, bem como com as normas e legislações pertinentes da nação. Baseado no princípio da legalidade, um administrador não poderia, por exemplo, realizar o monitoramento do correio eletrônico de um usuário sem o conhecimento deste. Tal ação fere diretamente os direitos fundamentais (sigilo) do cidadão, garantidos pela Constituição Federal;
- **Privacidade**: capacidade de um sistema de manter incógnito um usuário, impossibilitando a ligação direta da identidade do usuário com as ações por ele realizadas;
- **Auditoria**: capacidade do sistema de auditar tudo o que foi realizado pelos usuários, detectando fraudes ou tentativas de ataque.

Conexões de entrada: gerência e controle de acesso

Um serviço pode:

- Estar acessível para qualquer máquina.
- Estar acessível apenas para algumas máquinas.
- Não estar acessível.

Um sistema Linux é capaz de oferecer vários serviços através da rede. Para cada serviço há pelo menos um processo (um daemon) escutando em alguma porta TCP e/ou UDP. Dependendo do serviço e da máquina onde ele é oferecido, três tipos de situação podem ser desejadas:

- Que o serviço esteja acessível de qualquer local da internet;
- Que o serviço esteja disponível apenas para alguns hosts;
- Que o serviço não esteja acessível.

Para melhor ilustrar essas três situações, considere o exemplo:

Deseja-se oferecer um servidor web onde os usuários atualizem seus conteúdos via NFS. Em tal sistema, o serviço web deve ser acessado de qualquer local da internet; o serviço NFS deve estar acessível apenas para a intranet. Qualquer outro serviço deve estar desligado, como, por exemplo, mail, finger ou ident.

Figura 5.1
O sistema Linux: da intranet à internet.

Veremos, a seguir, como proceder em cada uma dessas três situações.

Iniciar e parar serviços

```
/etc/init.d/<serviço> start|stop
```

Como um serviço é iniciado durante o boot do sistema?

- Para cada run-level são executados os scripts do */etc/rcN.d.*
  - S20ssh (inicia o serviço SSH com "prioridade" 20).
  - K20ssh (interrompe o serviço SSH com "prioridade" 20).

Uma das regras mais importantes da segurança de redes de computadores é: não oferecer serviços de rede desnecessários. Ou seja, o administrador deve desabilitar todos os serviços de rede que não serão utilizados, já que tais serviços podem ser explorados por um atacante remoto.

Esses comandos param serviços em execução ou iniciam serviços de maneira manual. Após um reboot, pode ser que esses serviços sejam iniciados automaticamente.

Para parar um serviço em execução, basta utilizar os scripts de controle do serviço, em geral localizados em */etc/init.d*. Por exemplo:

Para parar o servidor web Apache:

```
# /etc/init.d/apache stop
```

Para iniciá-lo:

```
# /etc/init.d/apache start
```

A maioria das distribuições atuais de Linux utiliza a inicialização do sistema com base no System V. Nesse sistema, para cada nível de execução há um diretório */etc/rcN.d* que contém links para os scripts de controle dos serviços em */etc/init.d*. Cada link tem o formato "XYYserviço", onde:

- X pode ter o valor K, que indica que o serviço deve ser finalizado, ou S, indicando que o serviço deve ser iniciado;
- YY são dois dígitos e servem para indicar a prioridade com que os serviços serão iniciados.
- Pode-se manipular esses scripts ou utilizar programas ou scripts para isso, como o update-rc.d e rcconf (Debian), ntsysv (Red Hat) e chkconfig.

Figura 5.2
Utilizando o *rcconf*.

Caso não haja um link para um serviço específico, não haverá alteração no estado desse serviço ao entrar nesse nível de execução. Para iniciar ou não um serviço, é necessário manipular esses links na inicialização do sistema. Isso pode ser feito manualmente ou com o auxílio de scripts ou programas especiais, como, por exemplo, linuxconf, webmin, ntsysv, rcconf, chkconfig ou update-rc.d.

Para utilizar o *rcconf* em uma máquina Debian, é necessário instalá-lo:

```
#apt-get install rcconf
```

Daemon inetd

- Capaz de iniciar vários serviços de modo dinâmico.
- Configurado pelo inetd.conf.
- Formato das linhas:

```
<porta> <socket> <protocolo> <wait/nowait> \ <usuário[.grupo]>
  <programa> <opções do \ programa>
```

- Exemplo:

```
smtp stream tcp nowait mail /usr/sbin/exim \ exim -bs
```

O inetd é um daemon especial, porque, ao contrário dos outros, não oferece um serviço específico. Ele pode ser configurado para escutar em determinadas portas e iniciar outros programas quando uma conexão é iniciada em uma dessas portas.

Em geral, o inetd é utilizado em serviços com uma taxa de solicitação baixa, para evitar que os daemons ou processos desses serviços sejam mantidos em memória o tempo todo. Por uma questão de desempenho, não se recomenda a utilização do inetd nos serviços que recebem muitas solicitações em intervalos curtos de tempo, já que para cada solicitação o inetd atenderá à requisição e chamará o programa para atender ao serviço, criando uma carga adicional e desnecessária de trabalho.

O daemon inetd é configurado através do arquivo */etc/inetd.conf*. Cada linha desse arquivo (exceto as linhas vazias e de comentários) descreve um serviço e possui o seguinte formato:

Nos sistemas baseados na distribuição Debian, o administrador poderá utilizar o comando ***rcconf*** para selecionar os serviços que devem ser iniciados durante a inicialização do sistema. O padrão do ***rcconf*** é alterar os serviços apenas para o run-level atual (normalmente o run-level 3).

```
<porta> <tipo de socket> <protocolo> <wait/nowait> < usuário[.
grupo]> <programa> <opções do programa>
```

- O parâmetro <porta> indica em que porta esse serviço será executado; é possível usar um valor numérico ou o nome do serviço, desde que ele esteja descrito no arquivo */etc/services*;
- O <tipo de socket> pode ser stream, dgram, raw, rdm ou seqpacket. Em geral, é utilizado o tipo stream para serviços que usam o protocolo TCP e o dgram para serviços que utilizam o UDP;
- O campo "<protocol>" indica o tipo de protocolo de transporte;
- O campo seguinte pode conter os valores wait ou nowait e indica se o daemon inetd deve continuar escutando nessa porta ou se deve esperar a finalização da conexão em andamento;
- O inetd pode executar processos com permissões de outro usuário ou grupo;
- Por fim, os dois últimos campos indicam o programa que deve ser executado e as opções que devem ser passadas para ele.

O exemplo a seguir configura o serviço de e-mail (porta 25) com o programa exim:

```
smtp  stream tcp nowait mail /usr/sbin/exim exim -bs
```

Para desabilitar um serviço no inetd, basta remover ou comentar sua linha no arquivo */etc/inetd.conf* e reiniciar o inetd *(/etc/init.d/inetd restart*).

Veremos a seguir como controlar o acesso de máquinas a um serviço.

tcpwrapper

- Duas formas: daemon tcpd e biblioteca libwrap.
- tcpd + inetd.
- smtp stream tcp nowait mail \ /usr/sbin/tcpd /usr/sbin/exim exim -bs
- Daemon sshd: "linkado" com a libwrap.
- Controle de acesso é feito pelos arquivos *hosts.allow* e *hosts.deny.*
- *hosts.allow*:
  - Lista o que pode ser acessado e é consultado primeiro.
- *hosts.deny*:
  - Lista o que deve ser negado e é consultado após o *hosts.allow.*

Cada linha tem a forma:

- *<daemon>: <hosts / redes>*

Nos itens anteriores vimos apenas como executar ou não um serviço, mas não como selecionar quem pode ter acesso ao serviço. Para resolver esse problema, foi criado o tcpwrapper. Existem duas formas de utilização:

- O programa tcpd é chamado antes do programa que atenderá a requisição (por exemplo, no inetd);
- O tcpwrapper é utilizado através da compilação dos daemons "linkados" com a biblioteca libwrap.

Tomando o exemplo do item anterior, para controlar o acesso com o tcpwrapper ao serviço de e-mail utilizando o inetd, basta escrever a seguinte linha no inetd.conf:

```
smtp stream tcp nowait mail /usr/sbin/tcpd /usr/sbin/exim exim -bs
```

Para os daemons que não utilizam o inetd, mas são "linkados" com a libwrap, não é necessário utilizar o programa tcpd. Basta utilizar o comando *ldd </caminho/binario_daemon>* para verificar se um daemon foi "linkado" com a libwrap. O exemplo a seguir mostra que o daemon sshd foi "linkado" com a libwrap:

```
# ldd /usr/sbin/sshd

linux-gate.so.1 => (0x00156000)

libwrap.so.0 => /usr/lib/libwrap.so.0 (0x007c3000) <--- libpam.so.0
=> /lib/libpam.so.0 (0x007b9000)

libdl.so.2 => /lib/libdl.so.2 (0x007b3000)

[...]
```

Em ambos os casos de utilização (via libwrap ou tcpd), o controle de acesso aos serviços é configurado através dos arquivos */etc/hosts.allow* e */etc/hosts.deny*.

/etc/hosts.allow e hosts.deny

O arquivo */etc/hosts.allow* lista os hosts que podem acessar determinados serviços protegidos pelo *tcpwrapper*. Cada linha de configuração tem a forma:

```
<serviço>: <hosts e/ou redes>.
```

O exemplo a seguir permite o acesso ao daemon SSH para os hosts 192.168.0.1 e 10.0.0.2:

```
sshd: 192.168.0.1 10.0.0.2
```

De maneira análoga, o arquivo */etc/hosts.deny* nega o acesso de hosts a serviços e a sua sintaxe é igual à sintaxe do arquivo */etc/hosts.allow*.

Ambos os arquivos aceitam algumas palavras-chave com significados especiais, sendo as principais as palavras ALL, que significa qualquer serviço ou qualquer host, e EXCEPT, para excluir algum serviço ou algum host. Segue um exemplo utilizando essas palavras:

```
ALL: 127.0.0.1

ALL EXCEPT sshd: 192.168.0. 10.0.0.0/255.255.255.0 EXCEPT 10.0.0.1
```

Observe que foram utilizadas duas formas de designar redes: a primeira simplesmente terminando a rede com "."; e a segunda, usando uma máscara de rede.

O tcpwrapper consulta primeiro o arquivo */etc/hosts.allow* e, em seguida, o arquivo */etc/hosts.deny*. Para uma configuração bem restritiva, poderíamos colocar somente os serviços ou hosts desejados no arquivo *hosts.allow* e inserir uma linha "ALL: ALL" no hosts.deny.

Exemplo de configuração

```
sshd: 192.168.0.1 10.0.0.
```

Palavras-chaves especiais:

- ALL, EXCEPT
- ALL: 127.0.0.1
- ALL EXCEPT sshd: 10.0.0. EXCEPT 10.0.0.1
- ALL: ALL

Como liberar o SSH e negar todos os outros serviços?

Testando a configuração:

- *tcpdmatch* e *tcpdchk*
  - tcpdchk –v
  - tcpdmatch sshd 192.168.0.3

tcpdmatch e tcpdchk

Após efetuar alterações nos arquivos de controle do tcpwrapper, para verificar eventuais erros de configuração o administrador pode testar esses arquivos com o comando *tcpdchk*. Ao executar o comando *tcpdchk* com a opção *–v*, serão listadas as regras contidas nos arquivos de configuração e também avisos de erro, quando eles existirem.

Para testar o comportamento do *tcpwrapper*, o administrador de redes pode usar o comando *tcpdmatch*, que simula o que o *tcpwrapper* faria ao receber uma determinada conexão. Na sua forma mais básica, o comando espera dois argumentos: o daemon e o cliente da conexão. Por exemplo, para testar o que aconteceria ao tentar acessar o daemon SSH a partir do cliente 192.168.0.3, basta usar o seguinte comando:

```
# tcpdmatch sshd 192.168.0.3
```

Daemon xinetd

eXtended inetd: versão “estendida” do inetd:

- Já vem “linkado” com a libwrap (dispensa o uso do tcpd).
- Cada serviço pode ser configurado em um arquivo separado dentro do diretório */etc/xinet.d/*.

Sintaxe:

```
service <nome_serviço>
{
<atributos> = <valor> [<valor2>...]
...
}
```

O daemon xinetd (eXtended inetd) é uma espécie de daemon inetd mais poderoso. Ele já vem “linkado” com a libwrap e é capaz de prover controle de acesso aos serviços iniciados por ele. A forma de configuração também é diferente, embora ele possua um aplicativo para converter para o seu formato o arquivo *inetd.conf*.

A configuração dos serviços oferecidos pelo xinetd é feita através do arquivo */etc/xinetd.conf*. No entanto, algumas distribuições utilizam arquivos de configurações adicionais (um por serviço), pois isso facilita a criação de scritps que manipulam essas configurações de forma automática. Esses arquivos ficam no diretório */etc/xinet.d/* e são incluídos a partir da opção *includedir/etc/xinetd.d* do arquivo */etc/xinetd.conf*. A sintaxe para serviços iniciados pelo xinetd é:

```
service <nome_serviço>
{
<atributos> = <valor> [<valor2>...]
}
```

Lista de atributos, seus valores e significados:

- **socket_type** (valores stream, dgram, raw, seqpacket): tipo de socket de rede. O stream é usado para os serviços que utilizam o protocolo TCP, e dgram para os serviços que utilizam o UDP.
- **type** (valores RPC, INTERNAL, TCPMUX, TCPMUXPLUS, UNLISTED): tipo de serviço. O valor INTERNAL é utilizado para serviços providos pelo próprio xinetd, como por exemplo o serviço echo. UNLISTED deve ser usado quando o serviço não está listado em */etc/services*.
- **yser** (valor <user>): ao ser executado, o servidor assumirá as permissões de <usuário>.
- **server** (valor </caminho/programa>): especifica o programa que será executado para atender o serviço.
- **server_args** (valor <args>): argumentos passados ao programa definido em server.
- **wait** (valores yes, no): especifica se o serviço é single-threaded ou multi-threaded, ou seja, se é capaz de atender apenas a uma conexão por vez ou a várias ao mesmo tempo.
- **port** (valor <port>): especifica a porta do serviço. Se o serviço estiver listado em */etc/services*, o valor da porta deve ser o mesmo nos dois locais.
- **only_from** (valor <ip>): permite acesso apenas a partir de <ip>. Aceita os mesmos formatos dos arquivos *hosts.deny/hosts.allow*.
- **no_access** (valor <ip>): bloqueia o acesso a partir de <ip>.
- **log_on_success** (valores PID, HOST, USERID, EXIT, DURATION, TRAFFIC): registra os acessos bem-sucedidos ao serviço. Múltiplos valores podem ser utilizados, especificando a informação que deve ser logada.
- **log_on_failure** (valores HOST, USERID, ATTEMPT): registra as falhas de acesso a um serviço. Uma falha pode ocorrer devido a um controle de acesso ou a um problema com o serviço.
- **disable** (valores yes, no): desabilita (yes) ou habilita (no) o serviço.

Exemplo de configuração do serviço Telnet:

- Arquivo */etc/xinetd.d/telnet*:

```
service telnet
{
socket_type   = stream
wait       = no
user       = root
server      = /usr/sbin/in.telnet
log_on_failure  += USERID
disable     = no
}
```

O exemplo a seguir configura o serviço Telnet no xinetd (arquivo */etc/xinetd.d/telnet*):

```
service telnet
{
flags                   = REUSE
socket_type      = stream
wait                    = no
user                    = root
server           = /usr/sbin/in.telnetd
log_on_failure   += USERID
disable          = no
}
```

Secure shell (SSH)

Permite login remoto e cópia de arquivos entre máquinas. Vantagens:

- Autenticação com chaves criptográficas.
- Conexão criptografada.
- Pode ser utilizado para “tunelar” outras conexões.
  - Exemplo: X11.

Secure shell (SSH) é um programa que permite o login remoto e a cópia de arquivos de modo seguro. Para isso, usa criptografia e, opcionalmente, autenticação baseada em chaves públicas. O SSH substitui com vantagens os chamados R comandos, utilizados para acessar ou efetuar operações em máquinas Linux remotas. Entre eles, temos:

- **rsh**: executa um comando numa máquina remota;
- **rlogin**: utilizado para iniciar uma sessão em modo terminal, em outra máquina;
- **rcp**: utilizado para copiar arquivos entre uma máquina remota e a máquina local.

Além desses, o SSH pode substituir com vantagens o Telnet e, em muitos casos, até mesmo o FTP, já que todos esses programas utilizam protocolos não criptografados.

O SSH é composto pelo daemon sshd, pelo cliente SSH e por programas auxiliares, como o ssh-keygen, o ssh-add e o ssh-agent. Veremos, a seguir, como instalar, configurar e usar o SSH.

SSH pode substituir outros comandos ou protocolos.

- R* comandos:
  - rsh: execução de comandos remotos.
  - rlogin: shell remoto.
  - rcp: cópia de arquivos.
- FTP.
- Telnet.

Daemon.

- sshd.

Clientes:

- ssh, scp e sftp.
- Existem clientes para várias plataformas, inclusive Windows (winscp e putty).

Configurando o daemon sshd

Arquivo de configuração:

- /etc/ssh/sshd_config

Principais palavras-chave:

- ListenAddress <ip>
- Protocol<"1","2","2,1", "1,2">
- PermitRootLogin <yes|no>
- PermitEmptyPasswords <yes|no>
- X11Forwarding <yes|no>

O daemon sshd é configurado através do arquivo */etc/ssh/sshd_config*. Sua configuração padrão é suficiente para sua utilização. Porém, o administrador pode desejar alterar alguns parâmetros. Cada linha do arquivo de configuração contém uma palavra-chave seguida de um valor. Linhas comentadas iniciam com o caractere #. Principais palavras-chaves, valores possíveis e comentários:

- **ListenAddress** (<IP>): faz com que o daemon apenas escute no endereço configurado. Deve ser um endereço IP de uma interface ou o endereço 0.0.0.0 para escutar em todas as interfaces da máquina.
- **Protocol** (1; 2; 2,1; 1,2): versão do protocolo SSH. Dê preferência pela utilização do protocolo 2.
- **PermitRootLogin** (yes, no): permite ou não o login do usuário root. Alguns administradores não permitem o login de root remotamente.
- **PermitEmptyPasswords** (yes, no): permite o uso de senhas vazias. Usado principalmente para se logar com uma chave pública sem senha.
- **X11Forwarding** (yes, no): permite que o protocolo X11 seja tunelado via SSH, permitindo a execução remota de aplicações gráficas.

- **AllowTcpForwarding** (yes, no): permite o "tunelamento" de portas TCP para outros hosts.
- **AllowUsers** (<padrão>): se especificado, permite o acesso apenas aos usuários que combinem com os padrões configurados. Caracteres * e ? são curingas. Por exemplo: aluno* combina com os logins aluno1, alunoabc etc. O padrão root@host1 combina com o login root, feito a partir de host1.
- **PasswordAuthentication** (yes, no): permite o login com senhas (semelhante ao telnet).
- **PubkeyAuthentication** (yes, no): permite a autenticação com chaves públicas (aplica-se apenas na versão 2 do protocolo).
- **RSAAuthentication** (yes, no): permite autenticação com chaves do tipo RSA (apenas protocolo 1).
- **StrictModes** (yes, no): faz com que o daemon sshd verifique algumas configurações de segurança (por exemplo, permissões no $HOME do usuário).
- **UsePrivilegeSeparation** (yes, no): após a autenticação de um usuário, o sshd lança um processo-filho, para atender a conexão que tem apenas os privilégios do usuário conectado. Isso impede que um usuário malicioso explore o sshd para ganhar privilégios de root.
- **Subsystem** (<nome> </caminho/ programa>): configura um subsistema externo, como por exemplo o sftp-server (necessário para a utilização do comando *sftp*).

Principais palavras-chave:

- AllowTcpForwarding <yes|no>
- AllowUsers <padrão>
- PasswordAuthentication <yes|no>
- PubkeyAuthentication <yes|no>
- RSAAuthentication <yes|no>
- UsePrivilegeSeparation <yes|no>

Chaves públicas

Chaves compartilhadas x chaves públicas.

- A chave pública é mais segura? Por quê?
  - Protocolo SSH v1: RSA.
  - Protocolo SSH v2: RSA ou DSA.

Gerando uma chave RSA:

```
$ ssh-keygen -t rsa
  Generating public/private rsa key pair.
  Enter file in which to save the key\ (/home/usuario/.ssh/id_
  rsa): <enter para aceitar o\ padrão>
  Enter passphrase (empty for no passphrase): <senha>
  Enter same passphrase again: <senha>
  Your identification has been saved in\ /home/usuario/.ssh/id_rsa.
  Your public key has been saved in\ /home/usuario/.ssh/id_rsa.pub.
  The key fingerprint is:
  c8:fd:83:c3:f5:b1:e4:cd:4e:2b:a9:1a:29:af:d2:ed\ usuario@teste.com
  $ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
```

Cliente SSH

Formas de uso:

- ssh maquina
- ssh usuario@maquina
- ssh -i <chave> usuario@maquina
- ssh -X usuario@maquina
- ssh usuario@maquina <comando>

Uso de chaves públicas com o SSH

O SSH é capaz de utilizar chaves públicas para autenticação. Mas o que vem a ser isso? Em criptografia há, basicamente, dois tipos de chaves (ou segredos ou senhas):

- **No primeiro tipo**: uma única chave é utilizada para criptografar e descriptografar um conjunto de dados;
- **No segundo tipo**: chamado de chave assimétrica; é utilizado um par de chaves, onde uma chave apenas criptografa dados e a outra apenas descriptografa dados. Assim, se criptografamos um dado com a chave A, será necessário usar a chave B para descriptografar, e vice-versa. Uma dessas chaves é denominada chave pública, e pode ser distribuída à vontade; a outra é denominada chave privada, e deve ser mantida segura.

Na autenticação com chaves públicas, uma das pontas da comunicação (por exemplo, o servidor) possui a chave pública do usuário e a utiliza para criptografar um desafio. Esse desafio é enviado à outra ponta (o cliente), que deverá ser capaz de descriptografá-lo utilizando sua chave privada, para provar que ele é o dono da chave pública em questão.

Para utilizar chaves públicas com o SSH, primeiro é necessário gerar o seu par de chaves. Isso pode ser feito com o comando *ssh-keygen*. É necessário informar o tipo de chave que se deseja criar. Ou seja, RSA1 para o protocolo 1 do SSH, e RSA ou DSA para o protocolo 2. Exemplo:

```
$ ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/home/usuario/.ssh/id_rsa):
<enter para aceitar o padrão>

Enter passphrase (empty for no passphrase): <senha> Enter same
passphrase again: <senha>

Your identification has been saved in /home/usuario/.ssh/id_rsa.
Your public key has been saved in /home/usuario/.ssh/id_rsa.pub. The
key fingerprint is: c8:fd:83:c3:f5:b1:e4:cd:4e:2b:a9:1a:29:af:d2:ed
usuario@teste.com
```

Observe que o comando solicita uma frase-senha (passphrase). Por padrão, o comando gera suas chaves e as coloca no diretório *.ssh* do $HOME do usuário. O usuário deve, então, inserir sua chave pública (no exemplo anterior, está contida no arquivo *id_rsa.pub*) no arquivo *$HOME/.ssh/authorized_keys* dos hosts onde deseja se logar. Uma atenção especial deve ser dispensada às permissões do diretório *.ssh* e seus arquivos. O diretório e o arquivo que contêm a chave privada devem ter permissões de leitura e escrita apenas para o usuário.
O usuário deve copiar sua chave privada para os hosts onde pretende se conectar.

Gerando uma chave RSA, o uso do cliente SSH é bastante simples e será demonstrado em alguns exemplos.

Efetuando um login no host *teste.com* com o usuário-exemplo e usando a autenticação por senha:

```
$ ssh exemplo@teste.com exemplo@teste.com password: <senha>
```

Efetuando o login no host teste.com, utilizando uma chave pública e permitindo o tunelamento de conexões X11. A opção *-i* especifica o arquivo que contém a chave privada, e a opção -X habilita o X11 Forwarding:

```
$ ssh -i minhachave -X usuario@teste.com
```

SSH também pode ser utilizado de maneira não interativa. Basta acrescentar à linha de comando o(s) comando(s) que se deseja executar no host remoto. O exemplo a seguir executa o comando *ls –la* no host e finaliza a conexão:

```
$ ssh usuario@teste.com “ls -la”
```

SSH também pode ser utilizado para a cópia de arquivos. Nesse caso, é utilizado o comando *SCP*. A especificação do destino ou da origem remota assume o formato <usuario>@<host>:<caminho>. Para copiar o arquivo1 para */home/usuario do host exemplo.com*, é possível usar:

```
$ scp arquivo1 usuario@exemplo.com:/home/usuario
```

Para utilizar o SCP, é necessário conhecer previamente a localização do arquivo na máquina remota. Outra opção para a cópia de arquivos é o comando *sftp* (o servidor SSH deve suportar esse subsistema). O *sftp* funciona da mesma maneira que o comando *ftp*, mas com as vantagens do SSH, como a criptografia e a autenticação com chave pública. Após iniciar uma conexão com *sftp*, é possível usar os comandos *ls*, *cd*, *get*, *put*, *mkdir* de forma semelhante a uma sessão FTP.

O exemplo a seguir mostra o comando *sftp* sendo utilizado para a transferência de um arquivo para a máquina local (comando *get*) e, em seguida, para a transferência de um arquivo para a máquina remota (comando *put*):

```
$ sftp usuario@exemplo.com sftp> cd /tmp

sftp> get arquivo1

arquivo1 100% 1044 79.7KB/s 00:00 sftp> put arquivo2

arquivo2 100% 990 75.3KB/s 00:00 sftp> bye
```

Um recurso bastante poderoso do SSH é a possibilidade de se usar uma conexão segura para “tunelar” portas TCP. Isso permite utilizar conexões SSH para proteger conexões que, de outra forma, seriam inseguras. Imagine fazer o download de e-mails através do protocolo POP3 (que roda na porta 110 do servidor). Pode-se criar um túnel com o SSH da máquina cliente para o servidor com o comando:

```
$ ssh usuario@exemplo.com -L 1100:servidor:110
```

Com esse comando, qualquer conexão feita na porta 1100 do cliente seria “tunelada” para a porta 110 do servidor. Observe que se o servidor POP3 e a máquina para onde está sendo feita a conexão SSH forem os mesmos, é possível utilizar a opção -L 1100:localhost:110.

SSH também permite o "tunelamento" de portas ou conexões TCP:

```
ssh usuario@maquina -L <porta_local>:<host>:<porta>
  ssh usuario@maquina -L 1100:localhost:110
  O que isso faz? ssh usuario@maquina -L 3128:proxyweb:3128
```

Logs

- Contam a "história" do sistema.
- Contêm registros de eventos, avisos e erros.
- Auxiliam na detecção de problemas.
- São administrados pelo syslog.
  - Embora alguns daemons não utilizem o syslog.
- É necessário analisar logs se o sistema está funcionando corretamente? Por quê?

Os logs do sistema são importantes aliados do administrador de redes. Os logs contam a história do sistema. Neles encontramos eventos normais (e também avisos e erros) registrados pelo kernel e por outros daemons e programas. Os logs devem ser utilizados pelo administrador não só na resolução de problemas, mas também para acompanhar a saúde do sistema.

Principais arquivos:

- /var/log/messages
- /var/log/auth.log
- /var/log/daemon.log

Alguns arquivos são binários, como *lastlog* e *faillog.*

Utilize os comandos:

- last.
- lastlog.
- faillog.

Análise de arquivos de log

Em geral, os arquivos de log ficam no diretório */var/log*.

O número de arquivos e seu conteúdo variam de acordo com os daemons ou processos que a máquina executa e da configuração do programa syslog. No entanto, alguns programas podem criar seus arquivos de log diretamente (geralmente no arquivo */var/log*) sem o auxílio do syslog.

Os arquivos que devem merecer mais atenção do administrador variam de acordo com os serviços oferecidos pelo sistema. Há os que devem sempre receber atenção:

- **/var/log/messages**: contém mensagens diversas do sistema;
- **/var/log/auth.log**: contém informações relacionadas a autenticações;
- **/var/log/daemon.log**: contém informações de daemons diversos.

Em vez de texto, alguns arquivos dentro do */var/log* mantêm informações em modo binário, como os arquivos *lastlog* e *faillog*. Para a leitura do conteúdo desses arquivos, são utilizados

os comandos: *lastlog*, que lista o último login de cada usuário; *last*, que lista os últimos logins feitos no sistema; e *faillog*, que lista as tentativas malsucedidas de logins no sistema.

Ao analisar os logs, o administrador deve procurar por eventos que normalmente não deveriam ocorrer, como, por exemplo: logins de root vindos de máquinas estranhas e em horários que normalmente o administrador não logaria no sistema; daemons que morrem sem razão aparente; interfaces de rede colocadas em modo promíscuo.

A tarefa de análise de logs pode ser bastante complexa em um sistema que gera uma grande quantidade de logs. Para facilitar essa tarefa, o administrador pode escrever scripts que procurem por determinados padrões dentro dos arquivos, ou utilizar um dos muitos analisadores de log disponíveis, como o swatch (http://sourceforge.net/projects/swatch/) e o logcheck (http://sourceforge.net/projects/sentrytools/).

O que procurar nos logs?

- Logins remotos de máquinas desconhecidas.
- Logins em horários não usuais.
- Daemons "morrendo".
- Interfaces colocadas em modo promíscuo.
- Uploads anônimos no FTP.
- Entre outras informações.
- Logs remotos ou Logs hosts.

De fácil implementação:

- "syslogd -r" no log host.
- *<facility.priority> @<hosts>*
  - No arquivo */etc/syslog.conf* das outras máquinas.

Logs remotos

Além dos hosts, roteadores e até mesmo switches podem ser configurados para fazer o log remoto. A centralização dos logs facilita não só a segurança, mas a própria administração da rede.

Como os logs guardam muitas evidências de ações maliciosas de um ataque, é bastante comum que um atacante, após invadir uma máquina, apague ou altere os logs do sistema para esconder seus rastros. Para evitar isso, é possível configurar o sistema para que ele efetue os seus logs pela rede em um outro sistema. A máquina responsável por receber os logs através da rede é denominada loghost e, em geral, essa é sua única função.

A configuração básica para que o syslogd receba logs pela rede é feita acrescentando a opção *-r* ao comando de inicialização do daemon. Nos hosts que vão gerar os logs é necessário editar o arquivo de configuração do syslog. E para cada priority ou facility que se deseje registrar, deve-se acrescentar "@<log host>" à lista de ações.

O exemplo a seguir, uma linha do arquivo */etc/syslog.conf*, faz com que o daemon syslog registre os eventos de logins na máquina lhost.exemplo:

```
authpriv.* @lhost.exemplo
```

Observe que o serviço syslog deve ser reiniciado após uma alteração na configuração.

Analisadores de logs

- A atividade de ler logs é difícil, repetitiva e com poucos resultados visíveis.
- Necessidade de associar registros entre os diversos equipamentos da rede.
- Exemplos: logcheck, swatch e log-analysis.

Logcheck

Projetado para executar automaticamente e verificar o sistema de arquivos de log em busca de registros que permitam identificar violações de segurança e atividades incomuns.

- Instalação:

```
#apt-get install logcheck logcheck-database logtail
```

- Arquivo de configuração:
  - */etc/logcheck/logcheck.conf*
- Paranóico (paranoid).
- Servidor (server).
- Estação de trabalho (workstation).

Em muitos ambientes computacionais a tarefa de ler logs é um atividade difícil, repetitiva e com poucos resultados visíveis. Um análise aprofundada exige que o usuário possua grandes conhecimentos do sistema, pois os logs registram tudo que acontece com o kernel, com os daemons e demais utilitários. No arquivo */var/log/syslog*, por exemplo, são gravadas automaticamente entradas e saídas de usuários ou invasores, comandos efetuados, mensagens, etc.

Outro problema em acompanhar diariamente os logs está relacionado com a necessidade de associar registros entre os diversos equipamentos da rede, buscando, por exemplo, rastrear os passos de uma possível invasão. Um exemplo seria a necessidade de, ao verificar o log de um servidor web e encontrar um registro suspeito, ter de cruzar essa informação com o log de um roteador ou firewall. Infelizmente não existe uma norma que defina um padrão único e uniforme para o registro de logs em qualquer sistema ou aplicação. Dessa forma, cada desenvolvedor constrói o seu próprio padrão, dificultando, ainda mais, sua associação com outro registro.

Nesse cenário aparecem ferramentas como o binário **logcheck, que permite filtrar e associar** registros de forma simples e rápida. O **logcheck foi projetado para executar automa**ticamente e verificar o sistema de arquivos de log em busca de registros que permitam identificar violações de segurança e atividades incomuns. O logcheck utiliza um programa chamado logtail, que "lembra" da última posição lida (linha) em um arquivo de log. Dessa forma ele evita registrar um mesmo evento várias vezes.

Por padrão, o logcheck é executado de hora em hora ou após um reboot da máquina. Seu funcionamento consiste, basicamente, em comparar as linhas dos arquivos de log do sistema com sua base de informações. Quando uma linha possui um correspondente com a base de dados, uma mensagem é enviada para o administrador informando o problema.

Instalação

Para instalar o logcheck em um ambiente Debian, execute:

```
#apt-get install logcheck logcheck-database logtail
```

Configuração

As configurações do logcheck estão disponíveis no arquivo */etc/logcheck/logcheck.conf*. Os principais parâmetros são o usuário padrão para o qual serão enviados os relatórios de alerta (SENDMAILTO) e o nível de detalhe da análise (REPORTLEVEL). Os três níveis de detalhe possíveis são: workstation, server e paranoid. O nível server (servidor) é o nível padrão, o paranoid (paranóico) é recomendado apenas para máquinas de alta segurança executando poucos serviços e workstation (estação de trabalho), para máquinas relativamente não críticas. Se desejar adicionar novos arquivos de logs, adicione-os em */etc/logcheck/logcheck.logfiles*. Esse arquivo já está configurado para utilizar o syslog.

Para executar o logcheck, basta digitar no shell:

```
#su -s /bin/bash -c "/usr/sbin/logcheck" logcheck
```

O que o atacante faz após invadir uma máquina?

- Apagar logs.
- Instalar rootkits.
  - Modificam binários e/ou módulos do kernel.

Por mais atento que seja, nenhum administrador está totalmente livre de ter uma de suas máquinas invadidas. Embora raro, pode acontecer de um atacante explorar uma vulnerabilidade ainda não conhecida ou divulgada de algum daemon, para a qual ainda não há uma correção disponível.

Ao invadir um sistema, o atacante vai, inicialmente, tentar apagar seus rastros e garantir uma forma de controlar a máquina, para permitir que ele se logue novamente. Uma forma de fazer isso é através dos chamados rootkits, que são conjuntos de programas para esses objetivos. Em geral, os rootkits sobrescrevem binários do sistema (por *exemplo,* init*,* ls*,* netstat) e, em alguns casos, carregam módulos no kernel. A seguir serão apresentadas algumas ações que podem ser tomadas para detectar um sistema comprometido, além de dicas sobre como proceder no caso de o sistema ter sido invadido.

Detectando uma invasão

- Logs.
- Arquivos escondidos.
- Binários modificados (soma MD5).
- Arquivos de histórico de shell.
- Estado das interfaces.
- Tráfego de rede.

O grau de facilidade ou dificuldade em se detectar um sistema invadido vai depender da habilidade do atacante. Indícios de invasão podem ser obtidos: através dos logs do sistema; do arquivo de histórico do shell do root e/ou de usuários; de logs do firewall; de falhas (segmentation fault) em comandos do sistema; de arquivos escondidos (como binários dentro do */dev*); comparando a soma MD5 dos principais binários do sistema com os valores contidos nos arquivos de informações do pacote ao qual o binário pertence; e através de alguns programas específicos para esse fim, como o rkhunter, o tripwire, OSSEC, Snort e o samhain.

Baixe o rkhunter no site http://sourceforge.net

O tripwire, o OSSEC, o Snort e o samhain são utilizados para verificação da integridade de arquivos. Já o rkhunter é uma ferramenta que verifica uma série de vulnerabilidades do sistema em busca de indícios de invasão.

RootKit Hunter

Rootkit Hunter, normalmente chamado de RKH, é uma ferramenta de segurança para análise e monitoramento para sistemas POSIX. Ajuda na detecção de rootkits, malwares e aponta falhas nas práticas de segurança. Rootkits têm estruturas e arquivos que são similares às assinaturas dos vírus.

Também oferece alguns tipos de busca que podem ser de grande valia para o administrador. Uma dessas buscas é a busca por arquivos que tenham tido as suas propriedades alteradas. A melhor forma de trabalhar é fazer a instalação do rkh logo após a instalação do Sistema Operacional.

Rootkit Hunter não é uma ferramenta reativa, ele apenas informa as ameaças encontradas. A investigação e tomada de ações é função do Administrador de Sistemas. O rkh inclui:

- Documentação em cada distribuição, que também pode ser acessada on-line,
- Lista de discussão rkhunter-users;
- FAQ;
- Arquivo de configuração: */etc/rkhunter.conf* ou /usr/local/etc/rkhunter.conf.

Rkhunter verifica o sistema em busca de anomalias. Procedimentos ao se detectar uma invasão:

- Conter a ação do atacante (exemplo: regras de firewall).
- Levantar “provas”.
- Avaliar a extensão do que foi comprometido.
- Verificar o backup (pode estar comprometido também).
- Se for possível, reinstalar o sistema.
- Notificar um Centro de Segurança e Resposta a Incidentes (CSIRT).

Os procedimentos pós-invasão vão variar de acordo com a importância do sistema comprometido e com a quantidade de informações que se deseja levantar da máquina comprometida. Alguns sistemas podem ser simplesmente retirados de operação, enquanto outros devem permanecer em operação sem interrupção.

Caso deseje obter mais informações da máquina comprometida, uma série de passos é necessária:

- Salvar a lista de conexões abertas da máquina. Isso deve ser feito com um binário do netstat que não tenha sido comprometido;
- Impedir que o atacante tenha acesso ao sistema ou que o sistema seja utilizado para atacar outros sistemas. Isso pode ser feito tirando a máquina da rede ou criando regras no firewall que impeçam essas conexões.

Caso deseje obter o máximo de informações do sistema comprometido e usar essas informações como provas contra um invasor, é necessário seguir um conjunto de passos que vão além do objetivo desse item, mas que incluem: criação de imagens dos sistemas de arquivos comprometidos com somas de verificação MD5; imagens do swap e da memória da máquina etc. Todos esses passos devem ser rigorosamente documentados.

Para colocar o sistema novamente em operação, é recomendável que se faça uma nova instalação do sistema, já que pode ser difícil determinar exatamente o que foi comprometido e

instalar o rkhunter. Também devem ser verificados os backups da máquina, já que ela pode ter sido comprometida há tempos e os backups terem guardado binários comprometidos.

Por fim, se sua instituição possui um grupo de segurança, notifique-o da invasão. Caso contrário, notifique um Centro de Segurança e Resposta a Incidentes (CSIRT), que pode ser encontrado em "Informações de Contato de Grupos de Segurança Brasileiros" no site do CERT.br.

Procedimentos de detecção de invasão

Análise forense:

- Requer conhecimento especializado e procedimentos padronizados.
- Coleta de evidências:
- Imagens dos sistemas de arquivos.
- Imagens do swap e da memória.
- Lista de conexões abertas.
- Medidas preventivas úteis:
- Manter os relógios das máquinas sincronizados; a análise de logs depende de logs com horários corretos.
- Desabilitar serviços não utilizados e restringir o acesso a outros serviços que não devem ser públicos.
- Substituir programas ou protocolos sem criptografias (exemplo: SSH em vez de Telnet, https em vez de http).
- Manter backups das máquinas, principalmente das configurações e dados de usuários.
- Manter os sistemas atualizados.
- Análisar logs e sempre que possível concentrar os logs num loghost.
- Estabeler procedimentos de armazenamento e monitoramento dos logs.

Algumas medidas simples podem tornar um sistema mais seguro, como por exemplo:

- Manter os relógios das máquinas sincronizados. A análise de logs depende de informações com horários corretos. Uma dica é usar o ntpdate;
- Desabilitar serviços não utilizados e restringir o acesso a outros serviços que não devem ser públicos;
- Substituir programas e protocolos sem criptografia. Por exemplo, SSH em vez de Telnet; https em vez de http;
- Manter backups das máquinas, principalmente das configurações e dados de usuários;
- Executar periodicamente o rkhunter;
- Manter os sistemas atualizados;
- Manter-se sempre atualizado. Uma forma é participar das listas de discussão sobre os principais softwares e sistemas utilizados na rede;
- Analisar logs e, sempre que possível, concentrar os logs num loghost;
- Estabelecer procedimentos de armazenamento e monitoramento dos logs.

Leia a RFC 2196, que trata de políticas de segurança para redes conectadas à internet, e os documentos do NBSO no site do CERT.br, em "Práticas de Segurança para Administradores de Redes Internet" .

Finalmente, é importante ressaltar que o administrador deve se manter atualizado através de boletins, mailing lists e outras fontes de informação.

Roteiro de Atividades 5

Atividade 5.1 – Run Level

Descubra o atual nível de execução (run level) do sistema através do comando *runlevel*. Para esse run level, que daemons estão sendo executados e em que ordem eles são iniciados?

Atividade 5.2 – Run Level (continuação)

Interrompa o daemon sshd. Em seguida:

1. Remova-o da inicialização do sistema.
2. Configure a inicialização de modo que o daemon ssh seja iniciado com prioridade 60 nos run levels 2, 3, 4 e 5, e encerrado com prioridade 25 nos run levels 0, 1 e 6.
3. Reinicie o daemon sshd.

Atividade 5.3 – Xlnetd

Sobre o daemon xinetd:

1. Descubra quais serviços são iniciados por esse daemon.
2. Edite os arquivos de configuração do daemon xinetd, de modo que apenas o serviço echo continue ativo.

Atividade 5.4 – tcpwrapper

Sobre o tcpwrapper:

1. Configure o serviço telnetd no xinetd.
2. Restrinja o acesso ao serviço telnet, de modo que apenas o localhost (127.0.0.1) possa acessá-lo.
3. Permita que apenas o serviço sshd seja acessado pelo host 192.168.200.3 e pelo localhost (127.0.0.1). Verifique se o sshd foi "linkado" com a libwrap.

Atividade 5.5 – sshd

Configure o daemon sshd para que apenas o usuário root e mais um usuário (a sua escolha) possam efetuar logon. Teste sua configuração.

Atividade 5.6 – sshd: chave RSA

Crie uma chave RSA para o seu usuário e a utilize para se logar no sistema. Crie uma chave sem senha e configure o daemon sshd para que ele aceite esse tipo de login.

Atividade 5.7 – SCP

Crie um pequeno script que faça o backup de um diretório qualquer e o copie através do comando *scp* para o diretório */backup* de outra máquina. Utilize a chave RSA criada na atividade anterior.

Atividade 5.8 – Logs

Descubra quantas vezes o usuário root efetuou login, de acordo com o registrado nos logs atuais. Descubra, também, a que horas e de onde esses logins foram feitos.

Atividade 5.9 – syslogd

Configure o seu syslog para que ele aceite logs de outras máquinas. Configure, também, para que a facility auth seja logada na máquina de outro aluno ou equipe. Dica: combine essa atividade com outras equipes ou alunos.

Atividade 5.10 – Procurando por indícios de invasão

Procure por indícios de invasão na sua máquina. Para isso, verifique:

1. Se a interface de rede está em modo promíscuo; procure também nos logs;
2. Verifique a soma MD5 dos binários netstat e ifconfig. Dica: md5sum.
3. Verifique se o usuário root utilizou os comandos *ftp* e/ou *wget* para baixar algum arquivo. Dica: procure no ".bash_history".
4. Instale e execute o rkhunter.

Instalação do *rkh*:

```
# apt-get update

# apt-get install rkhunter
```

Execução do *rkh* (veja no *man* as diversas opções de execução). Lembrar sempre de fazer as atualizações da base de dados.

```
# rhkhunter --check ou

# rkhunter -c
```

O aluno troca o proprietário de um dos binários. Use o comando *kill*.

```
# chown aluno:aluno kill
```

Rode o *rkh* de novo e verifique os binários que foram adulterados.

```
# rkhunter -c
```

Não se esquecer de voltar com o *owner*:

```
# chown root:root kill
```

6

Segurança – Sistema Operacional

objetivos

Elaborar política de segurança utilizando como referência as Security Polices do SANs; entender os principais problemas de segurança antes, durante e depois da instalação de um sistema Linux; implementar configurações que aumentem sensivelmente o nível de segurança do Sistema Operacional Linux; instalar e configurar alguns aplicativos que buscam aumentar a segurança do Sistema Operacional e identificar possíveis ataques.

conceitos

Política de segurança; problemas de segurança do sistema Linux; configurações e aplicativos de segurança do Sistema Operacional Linux; identificação de ataques.

Procedimentos para instalação

Segundo o manual de segurança do Debian, são tarefas que devem ser realizadas para garantir um ambiente seguro:

- Decidir de quais serviços você necessita e limitá-los.
- Restringir usuários e permissões no sistema.
- Proteger os serviços oferecidos pelo sistema.
- Utilizar ferramentas para garantir que o uso desautorizado seja detectado.

As últimas décadas foram marcadas pela integração dos computadores no mundo e na vida das pessoas. Para isso, foram criados serviços baseados em nosso modelo social, o que permitiu a conexão entre redes financeiras, supermercados, lojas, clínicas, postos de saúde e até mesmo hospitais. Hoje, podemos comprar qualquer objeto pela rede de computadores, manipular informações bancárias, acessar diversos bancos de dados que contêm grande parte do conhecimento humano e, se necessário, até anunciar um produto pessoal em um site de trocas.

Tanta informatização e facilidade de comunicação trouxeram para as organizações uma nova preocupação: como anda a segurança de seus sistemas e informações neste novo mundo digital? Proteger-se contra invasores passou a ser um grande diferencial estratégico e de competitividade medido e acompanhado por clientes e fornecedores.

O processo de segurança de um ambiente computacional deve partir de um bom mecanismo de segurança física que consiga restringir e limitar ao máximo o acesso de pessoas estranhas ao ambiente. São exemplos de controles de segurança física o uso de circuitos internos de TV, guarda armada, divisão da empresa em perímetros, controles biométricos de acesso etc. Bons

controles físicos tendem a provocar uma "falsa sensação" de segurança, pois estão muito mais evidentes do que os controles lógicos implementados pela organização.

Neste Capítulo serão vistos os principais aspectos lógicos que devem ser considerados para que um sistema Debian possa ser classificado como "seguro". Observe que a palavra "seguro" encontra-se em destaque. Isso acontece porque, por mais esforço que se faça, nunca poderemos garantir que um sistema seja 100% seguro. As configurações e recomendações aqui descritas foram baseadas no manual de segurança do Debian e em um artigo de Antonio Cláudio Sales Pinheiro.

Antes de começar, é importante refletir sobre os serviços que serão disponibilizados pelo novo servidor que está sendo adicionado à rede interna da empresa. Essa reflexão torna-se necessária, já que grande parte dos controles e serviços a serem instalados dependem diretamente da destinação que será dada ao equipamento. Entre os pontos a serem analisados nesse momento encontram-se:

- Decidir de quais serviços você necessita e limitá-los;
- Restringir usuários e permissões no sistema;
- Proteger os serviços oferecidos pelo sistema;
- Utilizar ferramentas para garantir que o uso desautorizado seja detectado.

Configurando a BIOS e protegendo fisicamente o hardware.

Particionando o sistema:

- /tmp: não permitir que arquivos sejam executados a partir desse diretório.
- /var: colocar em partição separada.
- /etc, /bin e /sbin: montar em um CD-ROM.
- /home: ativar opção de quota.

Grande parte dos controles de segurança podem ser superados caso o atacante tenha acesso físico à máquina. Isso acontece porque um atacante poderia, simplesmente, carregar um Sistema Operacional utilizando uma unidade de mídia externa (CD, pendrive ou disquete) e, com isso, contornar qualquer restrição de permissão configurada no Linux. Na prática, apenas os arquivos cifrados dentro do disco estarão protegidos (desde que a chave não esteja no mesmo local).

Em um servidor de produção, é recomendado que o administrador remova qualquer possibilidade de realizar a carga do Sistema Operacional utilizando uma mídia externa. Para configurar essa opção é necessário acessar o Setup (normalmente é só apertar a tecla "del" após o sinal de post) e modificar a sequência de boot definindo o disco rígido como o primeiro mecanismo de carga. Para que o atacante não consiga modificar essa opção, é necessário proteger o sistema configurando, também, uma senha para acesso ao **BIOS**.

Se um atacante conseguir "roubar" o disco rígido do servidor e instalá-lo em outro equipamento, ele conseguirá remontar, sem problemas, os arquivos disponíveis no disco. Como os discos rígidos são hardwares pequenos, é provável que um atacante consiga retirá-lo da empresa sem ser notado, devendo o administrador, portanto, instalar trancas e cadeados no gabinete do servidor.

BIOS
Basic Input/Output System (Sistema Básico de Entrada/Saída). É um programa de computador pré-gravado em memória permanente (firmware) executado por um computador quando ligado. Ele é responsável pelo suporte básico de acesso ao hardware, bem como por iniciar a carga do Sistema Operacional.

Particionando o sistema

A escolha do modo como o disco rígido do servidor será dividido entre os diversos diretórios de um Linux dependerá, exclusivamente, dos serviços que esse servidor disponibilizará para a rede (exemplo: um servidor de arquivos teria, normalmente, grande parte do disco rígido destinado à partição */home*, enquanto um firewall possuirá uma partição */var* maior). Basicamente, os diretórios listados a seguir devem estar instalados em partições diferentes e com permissões próprias. Ao lado de cada diretório estão comentários que explicam as abordagens de acordo com os serviços disponibilizados.

- **/tmp**: responsável por armazenar os arquivos que serão manipulados temporariamente no ambiente. Na prática, qualquer usuário do sistema pode escrever arquivos nesse diretório, compilá-los e executá-los sem nenhum tipo de restrição.
- **/var**: responsável por armazenar os logs e registros do sistema. Um problema comum em máquinas Linux está associado ao fato de que, quando o espaço no disco termina (muitas vezes provocado pelo crescimento excessivo do log), o Sistema Operacional para de funcionar. Um atacante poderia provocar um ataque de negação de serviço simplesmente enviando uma grande quantidade de pacotes que fossem registrados pelo sistema. Separar o diretório */var* minimizará os ataques de negação de serviço, porém poderá dificultar a identificação do atacante, já que alguns logs poderão ser perdidos.
- **/etc**, **/bin**, **/sbin**: responsáveis, respectivamente, por armazenar os arquivos de configuração do ambiente, os binários de uso geral e os binários de uso exclusivo do administrador. Em sistemas onde exista uma grande necessidade de segurança, pode ser interessante armazenar e disponibilizar esses diretórios em um CD-ROM. Esse procedimento evitará que um atacante, ao comprometer um sistema, modifique algum arquivo de configuração ou binário que possa lhe garantir o retorno ao ambiente sem ser detectado.
- /**home**: responsável por armazenar os arquivos dos usuários. Qualquer diretório que um usuário possa escrever deve estar em uma partição separada e usar quotas de disco. Isso reduz o risco de um usuário "encher" o sistema de arquivos e provocar uma negação de serviço.

Regras importantes para o usuário root:

- Utilize uma senha de difícil dedução.
- Crie um usuário simples para o administrador do sistema e adicione-o ao grupo "wheel".
- Não execute o ambiente X ou qualquer outra aplicação de usuário como root.
- Nunca permita ao usuário root logar diretamente no sistema.
- Sempre use caminhos completos quando logado como root.
- Se um usuário só precisa rodar alguns comandos como root, considere usar o comando *sudo*.
- Nunca deixe o terminal quando você estiver logado como root.

O usuário root é o principal usuário do sistema e deve ser utilizado somente quando for estritamente necessário. Por padrão, esse usuário não possui nenhum tipo de restrição e, caso um atacante consiga acesso como root em um servidor, a única forma de confiar novamente no sistema é reinstalando-o completamente.

Quando falarmos em usuário root, devem ser seguidas algumas regras:

- Utilize uma senha de difícil dedução, que combine letras, números e caracteres especiais e tenha mais de 8 caracteres;
- Crie um usuário simples para o administrador do sistema e o adicione-o ao grupo "wheel" para que este possa se tornar root sempre que for necessário executar alguma atividade administrativa complexa;
- Não execute o ambiente X ou qualquer outra aplicação de usuário como root. Observe que se uma vulnerabilidade existir em uma aplicação executada com as permissões do usuário root e for explorada com sucesso, permitirá que sejam executados comandos sem nenhum tipo de controle ou restrição;
- Nunca permita ao usuário root logar diretamente no sistema;
- Sempre use caminhos completos quando logado como root (exemplo: o caminho completo do binário *ifconfig* é */sbin/ifconfig*). Se uma variável PATH for comprometida, o root pode acabar executando uma aplicação sem notar o comportamento suspeito do ambiente;
- Se um usuário só precisa rodar alguns comandos como root, considere usar o comando *sudo*;
- Nunca deixe o terminal quando você estiver logado como root.

Execute o mínimo de serviços

```
# rcconf
# dpkg -l
#apt-get remove
```

No capítulo anterior foi explicada a importância da remoção de todos os serviços desnecessários do servidor. Lembre-se, apenas, de que quanto maior for a quantidade de serviços, maior será o nível de risco. Afinal, um atacante pode explorar vulnerabilidades de um serviço para comprometer a segurança do Sistema Operacional.

Não se esqueça de desativar todos os serviços desnecessários utilizando o binário rcconf e, sempre que possível, substitua o inetd pelo xinetd.

Instale o mínimo de softwares

Removendo o compilador Perl:

```
# for i in /bin/* /sbin/* /usr/bin/* /usr/sbin/*; do [ -f $i ] && {
  type=`file $i | grep -il perl`; [ -n "$type" ] && echo $i; }; done
```

Ao realizar a instalação de um servidor Linux, selecione o modo de instalação mínima e adicione, apenas, os binários ou programas que você vai utilizar. Essa recomendação evitará que um atacante explore alguma vulnerabilidade em algum binário desnecessário e comprometa a segurança do Sistema Operacional.

Para listar todos os pacotes ou programas instalados em um Sistema Operacional Debian, execute:

```
# dpkg -l
```

Para remover um determinado pacote poderá ser utilizado o binário APT:

```
# apt-get remove <pacote>
```

Também devem ser removidos todos os compiladores instalados, evitando que o atacante possa compilar alguma aplicação na máquina.

Observe que em um sistema Debian existe um grande problema quando se tenta remover o compilador Perl, pois muitos binários dependem dessa ferramenta para funcionar. Antes de removê-lo, execute o script a seguir e verifique quais serão as aplicações que vão deixar de funcionar após a remoção.

```
# for i in /bin/* /sbin/* /usr/bin/* /usr/sbin/*; do [ -f $i ] && {
type=`file $i | grep -il perl`; [ -n "$type" ] && echo $i; }; done
```

Leia as listas de segurança do Debian (security mailing lists)

Mantenha-se sempre atualizado e saiba que os atacantes são "criaturas" que leem bastante. Assine boas listas de segurança e participe de fóruns sobre o assunto. Sempre que possível, se associe, também, a fóruns de hackers e crackers, e fique observando as estratégias discutidas para avaliar a melhor forma de proteção. A atividade de administração requer muita leitura e boa percepção para filtrar os conteúdos interessantes. Para ajudá-lo nessa difícil tarefa destacamos os seguintes fóruns, listas e sites:

- **Site do Grupo Perícia Forense**: http://www.guiatecnico.com.br/periciforense
- **Site SANS**: http://www.sans.org/
- **Site Módulo**: http://www.modulo.com.br
- **Security Focus**: http://www.securityfocus.com/
- **Lista CISSP-BR**: http://tech.groups.yahoo.com/group/cisspBR/
- **Lista Debian**: http://lists.debian.org
- **Grupo de Trabalho em Segurança de Redes**: http://gts.nic.br
- **Dragon Research Group**: http://dragonresearchgroup.org

Política de segurança

Toda implementação de novo serviço dentro de um ambiente deve possuir política aprovada pela alta administração.

A política de segurança de uma organização é um conjunto de documentos que desenham a forma como a rede e os serviços serão fornecidos. Nela é definido o que é considerado como uso aceitável dos recursos computacionais. Em alguns ambientes, a política pode ser utilizada para autorizar a organização e interceptar tráfego de rede, inspecionar o histórico de comandos executados pelos usuários e seus diretórios de pessoal. Essas atividades permitem diagnosticar problemas e servem como guia para as ações de auditorias. Quando realizadas sem o conhecimento dos usuários, podem gerar sérios problemas judiciais para a empresa e para o administrador.

Toda implementação de um novo serviço dentro de um ambiente deve possuir uma política associada, aprovada pela alta administração, documentada e divulgada plenamente.
A política assegura a tranquilidade que o administrador precisa para impor certas restrições e dificuldades de acesso ao ambiente, atendendo aos interesses da empresa. Quando criar políticas, lembre-se de que os textos devem ser fáceis de ler. Uma política de segurança deve, no mínimo, conter os seguintes assuntos:

- Uso aceitável;
- Proteções de tela e cuidados com senhas;
- Baixando e instalando software;
- Informações a respeito da monitoração dos usuários;
- Uso de software de antivírus;
- Cuidados com informações sigilosas (qualquer forma escrita, papel ou digital);
- Desligando o PC antes de sair;
- Uso de criptografia;
- Cuidados com laptops durante viagens e estadias em hotéis.

Diferentes usuários podem precisar de diferentes níveis ou tipos de acesso, e assim sua política pode variar para acomodar a todos.

Para ajudá-lo na elaboração das políticas de segurança, verifique alguns exemplos no site do SANS: http://www.sans.org/

Fortalecendo o sistema

Atualizações de segurança:

```
# apt-get update
# apt-get upgrade
```

Configuração da senha do Grub:

- Arquivo */boot/grub/ grub.cfg.*

Restringindo o uso do console:

- Remover do arquivo */etc/securetty* todas as linhas que contêm as palavras *vc* e *tty.*

Desativando a reinicialização do sistema com as teclas CTRL+ALT+DEL:

- Comente a seguinte linha no arquivo */etc/inittab*:

```
# ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

Nesse momento serão realizadas algumas configurações simples que permitem aumentar de forma significativa a segurança de um Sistema Operacional Linux Debian.

Atualizações de segurança

Uma das principais tarefas realizadas por um administrador é manter atualizados todos os sistemas operacionais da rede. Muitos dos problemas de segurança encontrados em um sistema operacional são corrigidos e disponibilizados para download pelo fabricante. No caso do Linux Debian, por se tratar de um software livre, existe um grupo especializado em discutir e fornecer pacotes contendo as correções para os problemas de segurança encontrados: http://security.debian.org.

Para atualizar o sistema, é necessário inserir a seguinte linha no arquivo */etc/apt/sources.list.* Por padrão, este link já está configurado na versão mais recente do Debian.

Para realizar a atualização, execute como root:

```
# apt-get update
# apt-get upgrade
```

Configurar senha do Grub

Boot loader é o primeiro programa a ser executado quando um computador é ligado. É o responsável por carregar e passar o controle para o kernel do SO escolhido (Debian, no nosso caso).

O Grub é um boot loader utilizado pelo Linux para gerenciar a carga de um ou vários sistemas operacionais em um mesmo computador. Ele oferece a possibilidade de uma pessoa não autorizada ter acesso à máquina, e através de um reboot alterar a senha de root, colocando o sistema em "single mode" usando apenas o parâmetro <*name-of-your-bootimage*> *init=/bin/sh* no aviso de boot. Este modo (single) permite controle total do sistema sem a exigência de senha.

Para gerar hash *password* para o Grub, deve ser utilizado o comando *grub-mkpasswd-pbkdf2.*

```
Enter password:

Reenter password:

Your PBKDF2 is

grub.pbkdf2.sha512.10000.86C2D42BDAE707F3C066EB74C809F299091DAE46F6B
519DEF16A74925D0546F6E51A46A11BEF697E0DE88A287334E206D112D48C9137938
B21098C6F4FC0378C.7A9F18E6A832091514368B3847EB767FE3EA9D0B5353894237
E2260F198EE5BA7FE40F395B9BA032AFCF328DE1588C727C493113889B31588B83F
73C19A1BB31

Esse string deverá ser colocado no final do arquivo /etc/grub.d/00_
header:

cat << EOF

set superusers="usuario"

password_pbkdf2 usuario grub.pbkdf2.sha512.10000.86C2D42BDAE707FC06
6EB74C809F299091DAE46F6B519DEF16A74925D0546F6E51A46A11BEF697E0DE88A2
87334E206D112D48C9137938B21098C6F4FC0378C.7A9F18E6A832091514368B3847
EB767FE3EA9D0B5353894237E2260F198EE5BA7FE40F395B9BA032AFCF328DE1588
C727C493113889B31588B83F73C19A1BB31

EOF
```

Salve o arquivo e execute o comando *update-grub,* que vai gerar um novo *grub.cfg*.

Restringindo o uso do console

O arquivo */etc/securetty* permite que você especifique em que dispositivos tty (terminais) o administrador (root) pode logar. Uma opção interessante é remover todas as linhas que contenham as palavras "vc" e "tty", certificando-se de que o root nunca faça login diretamente em um terminal. Para acessar o sistema como root será necessário conhecer a senha de um usuário comum (de preferência membro do grupo wheel) para depois promover a root usando o comando *su*.

Desativando a reinicialização do sistema

Um problema comum em máquinas Debian é a possibilidade de reiniciar o sistema utilizando as teclas "CTRL+ALT+DEL". Um usuário descuidado poderia reiniciar a máquina de forma intencional ou não.

Para remover essa capacidade, comente a seguinte linha no arquivo */etc/inittab*:

```
# ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

Montando partições.

- nosuid.
- noexec.
- nodev.

Uma das características interessantes que fazem do servidor Linux uma ótima opção para a instalação em servidores é sua capacidade de customização. No arquivo */etc/fstab* estão informados os principais pontos de montagem utilizados pelo sistema. Em uma instalação profissional, ou seja, uma instalação onde os diretórios */tmp*, */home* e */var* foram colocados em partições individuais, é possível customizar e remover alguns atributos evitando problemas de segurança relacionados a, por exemplo, ataques de *exploits*. Os seguintes atributos podem ser utilizados para montar qualquer partição do tipo ext2, ext3, ou reiserfs:

- **nosuid**: ignora o bit de SUID e trata-o como um arquivo normal;
- **noexec**: previne a execução de arquivos da partição;
- **nodev**: ignora dispositivos.

Infelizmente, esses ajustes podem ser facilmente contornados ao executar um caminho não direto (ver referências para mais informações). No entanto, se configurarmos */tmp* para noexec a maioria dos exploits desenhados para serem executados diretamente de */tmp* serão parados.

Exemplo do arquivo */etc/fstab*:

```
# /etc/fstab

# Created by anaconda on Mon Jan 14 11:55:25 2013

#

# Accessible filesystems, by reference, are maintained under '/dev/
disk'

# See man pages fstab(5), findfs(music), mount(music) and/or
blkid(music) for more info

#


UUID=2dc8462e-b6fc-47a4-8bbc-9e9a3fac6967 /  ext4 defaults 1 1

UUID=eecbc4a3-1fdb-4f93-9380-e2a18450538f /home  ext4 defaults 1 2

UUID=0b4f6b70-11db-4fbf-a3ff-b37a5f914fb9 swap  swap defaults 0 0

########################################

# /etc/fstab

# Created by anaconda on Mon Jan 14 11:55:25 2013
```

```
#
# Accessible filesystems, by reference, are maintained under '/dev/
disk'
# See man pages fstab( 5 ), findfs( 8 ), mount( 8 ) and/or blkid( 8
) for more info
#

UUID=2dc8462e-b6fc-47a4-8bbc-9e9a3fac6967 /  ext4 defaults 1 1
UUID=eecbc4a3-1fdb-4f93-9380-e2a18450538f /home  ext4 defaults 1 2
UUID=0b4f6b70-11db-4fbf-a3ff-b37a5f914fb9 swap  swap defaults 0 0
########################################
#
# /etc/fstab
# Created by anaconda on Mon Jan 14 11:55:25 2013
#
# Accessible filesystems, by reference, are maintained under '/dev/
disk'
# See man pages fstab( 5 ), findfs( 8 ), mount( 8 ) and/or blkid( 8
) for more info
#

UUID=2dc8462e-b6fc-47a4-8bbc-9e9a3fac6967 /  ext4 defaults 1 1
UUID=eecbc4a3-1fdb-4f93-9380-e2a18450538f /home  ext4 defaults 1 2
UUID=0b4f6b70-11db-4fbf-a3ff-b37a5f914fb9 swap  swap defaults 0 0
```

Colocar */tmp* no modo noexec pode impedir que certos scripts executem adequadamente.

Implementando quotas de disco

```
#apt-get install quota
```

- Modifique o arquivo */etc/fstab* e adicione *usrquota* e *grpquota* às partições que deseja ativar quota por usuário e quota por grupo.

```
/dev/sda6 /home ext3 noatime,nodev,nosuid,noexec,usrquota,grpquota 0 0
```

- Crie os arquivos *aquota.user* e *aquota.group* nas raízes das partições com suporte ativo para quota.
- Reinicie a máquina para os arquivos serem criados automaticamente.

Edite as quotas utilizando o comando *edquota*:

```
#edquota -u <usuario>
#edquota -g <grupo>
```

Colocar quotas em um sistema de arquivos restringe o uso do disco evitando que um único usuário possa consumir todo o espaço físico reservado a uma partição. As quotas podem ser implementadas com base em usuário ou grupo, e devem ser ativadas no kernel e adicionadas em um ponto de montagem dentro do arquivo */etc/fstab*.

Para instalar o suporte a quotas de disco do Debian, execute como root:

```
# apt-get install quota
```

Para ativar a quota, modifique o arquivo */etc/fstab* e adicione *usrquota* e *grpquota* às partições em que você quer restringir o uso de disco:

```
/dev/sda6 /home ext3 noatime,nodev,nosuid,noexec,usrquota,grpquota 0 0
```

Em cada partição que você ativar quotas, crie os arquivos de quota (*aquota.user* e *aquota.group*) e coloque-os na raiz da partição.

Exemplo de criação dos arquivos de quota para a partição */tmp*:

```
# touch /home/aquota.user
# touch /home/aquota.group
# chmod 600 /home/aquota.user
# chmod 600 /home/aquota.group
```

No Debian basta você adicionar a opção de quota no arquivo */etc/fstab* e reiniciar a máquina que os arquivos *aquota.user* e *aquota.group* serão criados automaticamente.

É interessante configurar o sistema para que, automaticamente, seja realizada uma verificação da política de quota implementada no servidor. Para realizar uma verificação semanal, utilizando o comando *#contrab –e*, adicione ao final do arquivo */etc/crontab*:

Listagem de código: adicionando verificação de quota ao *crontab*:

```
0 3 * * 0 /sbin/quotacheck -avugf.
```

Após reiniciar o servidor (# shutdown –r now), está na hora de configurar as quotas para usuários e grupos.

O seguinte comando pode ser utilizado para editar a quota do usuário kn:

```
# edquota -u kn
Disk quotas for user kn (uid 1000):
Filesystem blocks        soft          hard          inodes
soft      hard
/dev/sda6  10020         12000         13000         3
5  6
```

Explicando a função de cada coluna:

- **Filesystem**: partição que terá a quota do usuário ou grupo editada. No exemplo: */dev/sda6*;
- **blocks**: número máximo de blocos (especificado em Kbytes) que o usuário possui atualmente. No exemplo: 10020 Kbytes;
- **soft**: restrição mínima de espaço em disco usado. No exemplo: 12000 Kbytes;
- **hard**: limite máximo aceitável de uso em disco para o usuário/grupo editado. O sistema de quotas nunca deixará este limite ser ultrapassado. No exemplo: 13000 Kbytes;
- **inodes**: número máximo de arquivos (inodes) que o usuário possui atualmente na partição especificada;
- **soft**: restrição mínima de número de arquivos que o usuário ou grupo possui no disco;
- **hard**: restrição máxima de número de arquivos que o usuário ou grupo possui no disco.

A diferença entre o limite soft e hard é que, quando o limite soft é atingido, o usuário receberá um alerta (quota do usuário excedida) e, quando o hard é atingido, não será possível armazenar novos arquivos.

Para editar a conta do grupo kg, execute:

```
# edquota -g kg
```

Para mais detalhes, leia o manual do comando *edquota*.

Fornecendo acesso seguro ao usuário

- Autenticação do usuário: PAM.
- Garantir que o usuário root autentique apenas em terminais locais.
  - /etc/pam.d/login
- Restringir o uso de recursos por usuário/aplicação.
  - /etc/pam.d/lul-utem login

Plugglabe Authentication Modules (PAM) é um conjunto de bibliotecas que fornecem aos aplicativos Linux um mecanismo integrado de autenticação de usuários. O PAM possui uma série de arquivos de configuração dentro do diretório */etc/pam.d/*, que são utilizados para modificar seu comportamento. Por exemplo, o método utilizado para gerenciar sessões e definir o critério para criação de senhas (letras, números e caracteres especiais).Veremos a seguir algumas configurações.

Ativar o suporte MD5 nas aplicações de login e SSH

- /etc/pam.d/login
- /etc/pam.d/ssh

Para adicionar o suporte MD5 nas aplicações que utilizam o PAM, dificultando dessa forma as chances de um invasor descobrir alguma senha utilizando um ataque de força bruta, instale a biblioteca libpam-cracklib (#apt-get install libpam-cracklib) e realize as seguintes modificações no arquivo */etc/pam.d/common-password*:

```
# Comente a linha
# password required pam_unix.so nullok obscure min=4 max=8 md5
# Remova os comentários das linhas
```

Para carregar a *cracklib* e avaliar a qualidade de novas senhas, exigindo senhas com mais de 12 caracteres, além de criar um log sem permissão para a reutilização das últimas três senhas usadas:

```
password required pam_cracklib.so retry=3 minlen=6 difok=3 #
```

Para ativar o modo de autenticação utilizando o algoritmo MD5:

```
password required pam_unix.so use_authtok nullok md5 #
```

Antes de realizar essa alteração, tenha a certeza de instalar primeiro o libpam-cracklib ou então não será capaz de se logar no sistema.

O Cracklib depende do pacote wordlist (tal como wenglish, wspanish, wbritish). Portanto, tenha a certeza de instalar um que seja apropriado ao seu idioma ou o cracklib pode não ser totalmente útil (#apt-get install wbrazilian).

Garanta também que o usuário root autentique apenas em terminais locais; para isso, a seguinte linha deverá ser inserida no arquivo */etc/pam.d/common-auth*:

```
auth requisite pam_securetty.so
```

Lembre-se de restringir o uso do terminal para o usuário root e de restringir o uso de recursos por usuário ou aplicação.

No Linux é possível limitar, por exemplo, o número de logins concorrentes (de um determinado grupo de usuários ou de todo o sistema), número de processos, tamanho de memória etc. Essa capacidade pode ser ativada no arquivo */etc/pam.d/su*, removendo o comentário da seguinte linha:

```
session required pam_limits.so
```

Adiante será mostrado como configurar essas limitações para cada usuário ou aplicação.

Criando o grupo wheel

```
#groupadd wheel
#usermod -G wheel <usuario>
/etc/pam.d/su
```

Uma opção interessante é limitar a quantidade de usuários que podem se tornar root no sistema utilizando o comando *su*. Esse recurso torna-se interessante, pois um atacante, para conseguir o acesso como root, deverá também conhecer a senha de um usuário que esteja no grupo wheel. Para ativar essa opção é necessário adicionar o grupo wheel (#groupadd wheel) e adicionar algum usuário ao grupo wheel (#usermod –G wheel <usuario>). Remova o comentário da linha a seguir no arquivo */etc/pam.d/su* e realize as adaptações:

```
auth requisite pam_wheel.so group=wheel debug
```

Limitando o uso de recursos: o arquivo /etc/security/limits.conf

Os usuários do PAM podem limitar, para um determinado usuário e/ou aplicação, a quantidade de CPU, memória, pilhas etc. que o sistema pode fornecer. Esse tipo de comportamento impedirá que um determinado usuário possa consumir, de forma indiscriminada, todos os recursos do servidor provocando um ataque de negação de serviço. No entanto, ajustes muito restritos atrapalharão o funcionamento de seu sistema, provocando falhas em programas; por isso avalie com cuidado cada ajuste antes de implementá-lo.

Para saber mais, acesse "Entendendo e configurando o PAM"- www.ibm.com, "Securing Linux, Step by Step" - http://seifried.org e "Limiting and monitoring users" - http://seifried.org/lasg/users/

Pequeno exemplo de configuração do arquivo */etc/security/limits.conf*:

```
<domain>   <type> <item>        <value>

usuario    hard          nproc               10

@grupo     hard          nproc               5
```

Ações de login do usuário

Edite o arquivo */etc/login.defs*:

- FAILLOG_ENAB
- LOG_UNKFAIL_ENAB
- SYSLOG_SU_ENAB
- MD5_CRYPT_ENAB
- PASS_MAX_LEN

O arquivo */etc/login.defs* permite ativar, em uma máquina Linux, critérios de autenticação e senha definidos na política de segurança da organização. Observe que esse arquivo não é um arquivo de configuração do PAM e, por consequência, só vale para programas que utilizem os binários login ou su. Alguns parâmetros importantes que compõem esse arquivo:

```
FAILLOG_ENAB yes
```

Registra, no syslog, as falhas de login sempre que o usuário erra a senha.

```
LOG_UNKFAIL_ENAB yes
```

Registra, no syslog, as falhas de login causadas por usuário não cadastrado no sistema. Observe que, em alguns casos, o usuário poderia trocar o nome de login pela senha. Sempre que ativar essa opção, lembre-se de verificar se os arquivos de logs estão com permissões restritivas (640).

```
SYSLOG_SU_ENAB yes
```

Registra as tentativas de uso de binário su no syslog.

```
MD5_CRYPT_ENAB yes
```

Ativa o sistema de hash MD5 para uso nas senhas. Essa opção só deve ser ativada quando o PAM já estiver configurado para usar MD5.

```
PASS_MAX_LEN 50
```

Caso o uso de senhas MD5 já esteja ativado na configuração do PAM, essa variável deve ser setada com o mesmo valor utilizado no arquivo */etc/pam.d/login*.

Usando o sudo

```
#apt-get install sudo
```

Desativando acesso administrativo remoto:

- Arquivo */etc/security/access.conf*

Configurando o *umask:*

- # umask 022
- Convertendo 022 em binário 0 2 2
- 000 010 010
- Associando com as permissões rwx rwx rwx
- ==========
- Resultado rwx r-x r-x

Algumas vezes é necessário permitir que um usuário comum possa executar algumas tarefas administrativas (exemplo: reiniciar serviços ou o servidor). O binário sudo auxilia nesse processo, permitindo que um usuário qualquer execute comandos utilizando a identidade de outro usuário. Para instalar o sudo, execute como root:

```
# apt-get install sudo
```

O arquivo de configuração do sudo é */etc/sudoers*. Nesse arquivo deve ser inserido o nome do usuário comum e o comando que ele pode executar. Exemplo:

```
ROOT  ALL = (ALL) ALL
<usuario> ALL = NOPASSWD: /sbin/shutdown, /sbin/poweroff, /sbin/
halt, /sbin/reboot, /bin/cdrecord, /usr/bin/mount, /sur/bin/smbmount
```

O parâmetro NOPASSWD elimina a necessidade de o usuário digitar a senha do root. Avalie com cuidado essa opção.

Desativação de acesso administrativo remoto

Uma forma de evitar que logins remotos possam ser realizados utilizando a conta do root é modificar o arquivo */etc/security/access.conf*. Dessa forma, se um usuário desejar acessar o sistema como root, deverá realizar uma autenticação com um usuário comum e depois se promover utilizando o *su* ou executando a aplicação com o sudo. Para configurar essa opção adicione a seguinte linha no arquivo */etc/security/access.conf* (no arquivo padrão essa linha está comentada):

```
-:wheel:ALL EXCEPT LOCAL
```

Lembre-se de ativar o módulo pam_access para cada serviço (ou configuração padrão) em */etc/pam.d/* se quiser que suas alterações em */etc/security/access.conf* sejam respeitadas.

Configurando umask

O binário umask é utilizado para definir a máscara da permissão padrão a ser utilizada quando o usuário criar novos arquivos e/ou diretórios. Para verificar a máscara atual utilizada pelo sistema, digite:

```
# umask
```

Para modificar a máscara, basta digitar o comando acompanhado da nova máscara:

```
# umask 222
```

Embora o formato de permissão seja semelhante ao do binário chmod, o *umask* possui uma fórmula especial para calcular a permissão que será atribuída a novos arquivos e diretórios. Para descobrir como o comando *umask* funciona, você deve observar os bits da máscara, e não somente o número. Observe o exemplo:

```
# umask 022
Convertendo 022 em binário                    0      2      2
                                              000    010    010
Associando com as permissões dono, grupo e resto rwx    rwx    rwx
                                                     ==========
Resultado                                     rwx    r-x    r-x
```

Note que o bit 1 indica que a permissão não está configurada.

Na máscara anterior, todos os novos diretórios criados terão a permissão de leitura, escrita e execução para o dono; leitura e execução para o grupo e leitura e execução para o resto. Para novos arquivos existirá uma pequena diferença, pois, por questões de proteção, novos arquivos nunca terão a permissão de execução definida por padrão. Se você quiser definir um arquivo executável, deverá fazê-lo através do comando *chmod*.

Para modificar o *umask* de todos os usuários, edite o arquivo */etc/profile* e modifique esse parâmetro no final do arquivo.

Limitando o conteúdo que os usuários podem ver e acessar

Em alguns cenários pode ser necessário que usuários comuns possam autenticar no sistema e usar um interpretador de comandos (shell). Nesse caso é importante avaliar o Sistema de Arquivos em sua verificação dos diretórios e arquivos que poderiam ser lidos por um usuário qualquer.

Por padrão, um sistema Debian limita a visualização de arquivos considerados importantes do ponto de vista da segurança (exemplo: */etc/shadow*). Embora adequado, esse comportamento pode criar uma falsa sensação de segurança. Para avaliar os arquivos que podem ser lidos por usuários comuns, execute:

```
# find / -type f -a -perm +006 2>/dev/null

# find / -type d -a -perm +007 2>/dev/null
```

Limitando o acesso a outras informações de usuários

Se você permite acesso ao shell para os usuários comuns, pode ser interessante limitar a quantidade de informações que eles podem acessar de outros usuários. No Linux Debian, os diretórios de usuários em $HOME são criados com permissões 0755 (lidos pelo grupo e por todos). Isso permite que um usuário comum possa entrar no diretório de outros usuários e ler seus arquivos. Para alterar esse comportamento, altere DIR_MODE no arquivo de configuração */etc/adduser.conf* para 0750 (sem acesso de leitura para todos).

Note que a desativação de leitura para todos no diretório HOME do usuário evitará que o servidor web leia os diretórios de páginas pessoais *~/public_html*. Se deseja permitir aos usuários a publicação de páginas HTML em seus diretórios *~/public_html*, altere DIR_MODE para 0751.

Restringindo o acesso de usuários

Ao adicionar um novo usuário, por padrão, o Linux definirá um shell válido para que o usuário possa autenticar e utilizar o servidor. Acontece que em alguns casos esse recurso não é necessário e os usuários criados serão utilizados apenas para autenticação nos serviços disponibilizados (serviço de mensagens pop3 ou FTP, por exemplo). Para evitar que um usuário possa acessar um terminal em um servidor, basta definir um shell nulo (/dev/null) no arquivo */etc/passwd*. Exemplo:

```
<usuario>:x:1003:1004:,,,:/home/<usuario>:/bin/false
```

Logout de usuários ociosos

Um usuário descuidado pode se levantar de sua cadeira e acabar deixando a estação de trabalho autenticada com uma seção válida e ativa. Essa falha, embora momentânea, poderá comprometer a segurança do ambiente, já que um intruso com acesso físico à organização poderia ter acesso ao servidor simplesmente "sentando na frente da máquina autenticada". Uma forma de definir o timeout para seções inativas é adicionar ao arquivo */etc/profile* a variável TMOUT configurada com o tempo em segundos. Exemplo:

```
TMOUT=60 # para um minuto

export TMOUT
```

Segurança do kernel

No repositório Debian GNU/Linux são disponibilizados diversos patches para o kernel, aumentando sensivelmente a segurança do Sistema Operacional. Avalie com cuidado e teste exaustivamente a implementação de cada um antes de utilizá-los, pois algumas aplicações podem deixar de funcionar.

- **NSA Enhanced Linux (no pacote setools)**: ferramentas para aumentar a segurança do Linux;
- **kernel-patch-grsecurity2**: implementa controle de acesso mandatário, oferece proteção contra estouro de buffer, ACLs, "network randomness" (para tornar "OS fingerprint" mais difícil), entre outras características;
- **Kernel-patch-scripts**: scripts utilizados para corrigir problemas no kernel;
- **Kernel-patch-exec-shield**: proteção contra ataques do tipo smashing, entre outros tipos de ataque;
- **Suporte ao kernel IPSec (no pacote *kernel-patch-freeswan*)**: para uso com o protocolo IPSec.

Proteção da Libsafe

A biblioteca Libsafe, quando instalada, evita que alguns ataques do tipo buffer overflow sejam realizados com sucesso. Para carregar globalmente essa biblioteca execute, como root:

```
#apt-get install libsafe-hole-perl.
```

Algumas aplicações podem não funcionar após a instalação dessa biblioteca (em geral todas aquelas que fazem uso da *libc5*).

Removendo funcionalidades

Uma característica interessante no Linux, mas pouco utilizada, é a capacidade de customização do kernel do Sistema Operacional removendo o suporte de drivers de hardwares não instalados no servidor. Como regra básica, na hora de configurar o kernel, remova tudo o que você não precisar. Isso não só criará um kernel menor, mas também poderá remover as vulnerabilidades que podem residir dentro de drivers e outras funções.

Essa tarefa não será realizada nesse curso por se tratar de uma atividade demorada e que exige um bom conhecimento de hardware por parte do administrador. Mais informações podem ser encontradas nas referências bibliográficas.

Sistema de arquivos proc

Muitas características do kernel podem ser modificadas com comandos *echo* no sistema de arquivos */proc* ou usando o arquivo */etc/sysctl*. O script a seguir mostra e comenta algumas dessas características:

```
# proteção contra broadcast de ECHO (evita ataques do tipo smurf)

echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Evita que um máquina com duas placas de rede (uma interna e outra
externa) seja utilizada como roteador por um usuário malicioso.

echo 0 > /proc/sys/net/ipv4/ip_forward

# Proteção contra syn cookies ativada

echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# Registra pacotes estranhos # (isto inclui pacotes falsos, pacotes
com a rota de origem alterada e pacotes redirecionados)

echo 1 > /proc/sys/net/ipv4/conf/all/log_martians

# opção de desfragmentação sempre

echo 1 > /proc/sys/net/ipv4/ip_always_defrag

# proteção ativada contra mensagens ICMP

echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

# proteção contra ip spoofing

echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter

# Não aceita redirecionamento de ICMP

echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects

echo 0 > /proc/sys/net/ipv4/conf/all/send_redirects

# Desativa pacotes com rota de origem

echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route

echo 1 > /proc/sys/net/ipv4/conf/all/log_martians
```

Não se esqueça de colocar todas essas opções em um arquivo que seja executado sempre que o sistema for reiniciado. Todas essas alterações serão apagadas caso essa tarefa não seja realizada.

Uma forma simples de resolver esse problema é utilizar o arquivo */etc/sysctl.conf*. Sua configuração é bastante simples; basta tirar o */proc/sys/* dos caminhos mencionados anteriormente e substituir "/" por ".". Exemplo:

```
/bin/echo "0" > /proc/sys/net/ipv4/ip_forward

(Automaticamente em sysctl.conf:)

net.ipv4.ip_forward = 0
```

Fazendo uma cópia de segurança do sistema

Após realizar todas as configurações e testes no servidor e antes de colocá-lo em produção, é interessante que seja realizada uma cópia de segurança externa de todo o sistema (principalmente do diretório */etc*). Essa cópia deve ser realizada, preferencialmente em CD ou DVD e deverá ser utilizada sempre que o sistema for comprometido. Existem ferramentas interessantes que realizam, inclusive, a cópia binária do sistema. Um exemplo é o *ghost* ou o binário *dd*.

A seguir veremos algumas ferramentas de segurança.

chroot

Instalando e configurando o *chroot*:

```
# apt-get install chrootuid

# ldd /bin/bash e # ldd /bin/ls
```

- Copiar binários e bibliotecas.

Iniciando e testando o funcionamento.

```
# chroot /chroot/bash /bin/bash
```

O chroot permite limitar o ambiente de um determinado usuário ou serviço modificando o seu diretório raiz (/). Essa configuração envia o usuário para um ambiente controlado onde ele terá acesso somente a arquivos e executáveis definidos pelo administrador.

Instalação e configuração

Para ativar o chroot no Debian, execute:

```
# apt-get install chrootuid
```

Vamos criar um exemplo de como funciona o aplicativo chroot. Neste exemplo, o usuário será direcionado para um ambiente shell (bash) onde existirá apenas o comando *ls* e nada mais.

Antes de iniciarmos, temos de descobrir as bibliotecas dinâmicas que foram utilizadas para compilar os binários */bin/bash* e */bin/ls*. Para isso será utilizado o binário ldd. Execute:

```
# ldd /bin/bash

# ldd /bin/ls
```

Vamos agora criar os diretórios e copiar os binários e bibliotecas necessárias:

```
# mkdir -p /chroot/bash/bin
# mkdir -p /chroot/bash/lib
# cp /bin/bash /chroot/bash/bin
# cp /bin/ls /chroot/bash/bin
# cp /lib/libncurses.so.5 /chroot/bash/lib
# cp /lib/libdl.so.2 /chroot/bash/lib
# cp /lib/ld-linux.so.2 /chroot/bash/lib
# cp /lib/librt.so.1 /chroot/bash/lib
# cp /lib/libacl.so.1 /chroot/bash/lib
# cp /lib/libselinux.so.1 /chroot/bash/lib
# cp /lib/libattr.so.1 /chroot/bash/lib
# cp /lib/libsepol.so.1 /chroot/bash/lib
# cp /lib/libpthread.so.0 /chroot/bash/lib
```

Iniciando e testando o funcionamento

Para ativar o chroot, execute:

```
# chroot /chroot/bash /bin/bash
```

Se você receber um prompt dizendo "/", ele está funcionando. Caso contrário, ele dirá que algum arquivo está faltando. Lembre-se de que algumas bibliotecas compartilhadas dependem umas da outras. Observe que quando estiver dentro do chroot nada funcionará, exceto talvez o echo e o comando *cd*.

PortsentryPermite identificar quando um invasor está realizando um *port scanning* no servidor. Instalação e configuração:

```
# apt-get install portsentry
```

- Arquivo */etc/portsentry.conf*

Iniciando e testando o serviço:

```
# /etc/init.d/portsentry restart
```

O aplicativo portsentry permite identificar quando um invasor está realizando um port scanning no servidor. Para realizar seu trabalho ele abre várias portas, de acordo com seu arquivo de configuração, e monitora as tentativas de acesso a essas portas. Ao identificar uma tentativa de acesso às portas por ele monitoradas, o portsentry pode bloquear o endereço IP do atacante e enviar, inclusive, uma mensagem dizendo que ele foi detectado. Avalie com cuidado essa opção para não acabar negando acesso para máquinas válidas.

Instalação e configuração

Para instalar o portsentry, execute como root:

```
# apt-get install portsentry
```

O arquivo de configuração do portsentry (*/etc/portsentry.conf*) é dividido em seções, sendo que a principal refere-se às portas que ele vai monitorar. Observe, logo a seguir, que existem três tipos de linhas TCP_PORTS e UDP_PORTS. O modelo mais utilizado é o que não está comentado.

```
# Un-comment these if you are really anal:

#TCP_POR
TS="1,7,9,11,15,70,79,80,109,110,111,119,138,139,143,512,513,[...]"

#UDP_PORTS="1,7,9,66,67,68,69,111,137,138,161,162,474,513,517,518,[...]"

# Use these if you just want to be aware:

TCP_PORTS="1,11,15,79,111,119,143,540,635,1080,1524,[...]"

UDP_PORTS="1,7,9,69,161,162,513,635,640,641,700,37444,[...]"

# Use these for just bare-bones

#TCP_PORTS="1,11,15,110,111,143,540,635,1080,1524,2000,[...]"

#UDP_PORTS="1,7,9,69,161,162,513,640,700,32770,32771,[...]"
```

Não se esqueça de remover do arquivo as portas de serviços válidos instalados no servidor.

Agora vamos definir as ações que serão realizadas pelo portsentry quando detectar um atacante realizando um port scanning. Essa configuração é realizada na seção Dropping Routes:

```
# iptables support for Linux

KILL_ROUTE="/usr/local/bin/iptables -I INPUT -s $TARGET$ -j DROP"
```

Remova o comentário da linha KILL_ROUTE referente ao suporte a iptables. Lembre-se de que caso uma máquina válida tente acessar um serviço inválido disponibilizado pelo portsentry, ela será automaticamente bloqueada para qualquer serviço válido.

Caso deseje, é possível enviar uma mensagem de aviso para o intruso. Para habilitá-la, basta retirar o comentário da linha:

```
PORT_BANNER="** ACESSO PROIBIDO *** INTRUSION DETECTION SYSTEM"
```

Para finalizar, é possível configurar o portsentry para realizar um contra-ataque. Aqui vale a sua imaginação. Observe que pode ser executado um script qualquer:

```
KILL_RUN_CMD="/some/path/here/script $TARGET$ $PORT$"
```

Iniciando e testando o serviço

Para reiniciar o portsentry execute:

```
# /etc/init.d/portsentry restart
```

Para testar o funcionamento do portsentry, a partir de uma máquina remota tente se conectar a uma porta inválida. Exemplo:

```
# telnet 192.168.1.2 1080

Trying 192.1.1.2...
```

```
Connected to 192.1.1.2

Escape character is ?^ ]?

.****Acesso Proibido **** Intrusion Detection System
```

Se foram realizadas as configurações descritas, serão feitas as seguintes atividades pelo portsentry:

- Serão registradas em "portsentry.blocked.tcp" algumas informações do atacante. Vale lembrar que esse arquivo tem a mesma função do "hosts.deny", porém fornece uma saída mais detalhada.
- Posteriormente o portsentry fará o iptables negar as conexões vindas da máquina remota.

Tripwire

Ferramenta desenvolvida para monitoramento das modificações ocorridas no sistema de arquivos.

```
#apt-get install tripwire
```

- Editando e gerando o arquivo de configuração.
- Editando e gerando o arquivo de políticas.
- Criando a base de dados inicial.

Iniciando e testando o serviço:

```
#adduser marcelo

# tripwire --check
```

Integridade do sistema de arquivos com Tripwire

O Tripwire é um monitor de integridade para sistemas Unix desenvolvido por Gene Kim com a orientação de Gene Spafford (diretor do projeto na Universidade de Purdue). Ele usa diversas rotinas de "checksum/message-digest /secure-hash /signature" para detectar alterações nos arquivos, bem como para monitorar determinados itens da informação mantida pelo sistema. A configuração do Tripwire permite ao administrador especificar facilmente os arquivos e diretórios a serem monitorados e a especificar arquivos para os quais são permitidas alterações limitadas, sem gerar um aviso de alerta.

Instalação e configuração

Ao instalar o Tripwire é criado um banco de dados com informações dos arquivos que devem ser por ele monitorados. Esse banco de informações é armazenado de forma cifrada utilizando uma passphrase. Para instalar o Tripwire no Debian, execute:

```
#apt-get install tripwire
```

Serão realizadas algumas perguntas pelo Debconf (veja as respostas default):

- Você deseja criar a chave do site durante a instalação? *Sim*
- Você deseja criar a chave local durante a instalação? *Sim*
- Você deseja reconstruir o arquivo de configuração do Tripwire? *Sim*
- Você deseja reconstruir o arquivo de políticas do Tripwire? *Sim*
- Informe a senha para a chave do site: 123456

- Repita a senha: 123456
- Informe a senha para a chave local: 123456
- Repita a senha: 123456

Editando e gerando o arquivo de configuração

O arquivo */etc/tripwire/twcfg.txt* é um arquivo texto comum que lista as principais opções e atributos dessa ferramenta. Após realizar uma leitura e posterior customização desse arquivo, será necessário gerar o arquivo */etc/tripwire/tw.cfg* para o Tripwire com as novas configurações; para isso, usaremos o comando:

```
# twadmin --create-cfgfile --site-keyfile /etc/tripwire/site.key
twcfg.txt
```

É aconselhável que após a geração do arquivo */etc/tripwire/tw.cfg* seja apagado o arquivo texto.

Editando e gerando o arquivo de políticas

O arquivo */etc/tripwire/twpol.txt* contém a lista de diretórios e arquivos que serão monitorados pelo Tripwire, bem como as políticas e regras utilizadas para o monitoramento dos arquivos. Assim como o arquivo de configuração, é necessário cifrar esse arquivo e gerar o arquivo */etc/tripwire/tw.pol* para o Tripwire com a nova política a ser usada nas checagens. Para isso usaremos o comando:

```
# twadmin --create-polfile /etc/tripwire/twpol.txt
```

Da mesma forma que o arquivo texto usado para criar o arquivo de configurações, devemos apagar também o arquivos de texto usados para a geração das políticas, pelas mesmas razões.

Criando a base de dados inicial

O comando a seguir poderá ser utilizado para construir a base de dados:

```
# tripwire --init
```

Iniciando e testando o serviço

Após a construção da base de dados inicial, o Tripwire estará preparado para realizar checagens de integridade. Essa verificação é feita através da verificação do estado atual do sistema de arquivos e da comparação com os dados armazenados na base de dados. Dessa forma, o Tripwire poderá ser capaz de determinar alguma modificação sofrida pelos arquivos monitorados. A verificação de integridade pode ser feita com o comando:

```
# tripwire --check
```

Para testar o funcionamento, basta adicionar um novo usuário e realizar, novamente, a verificação de integridade.

Roteiro de Atividades 6

Atividade 6.1 – Elaborando uma política de segurança

Utilizando como referência a política de segurança disponível no site do SANS sobre o uso aceitável dos recursos computacionais (http://www.sans.org/resources/policies/Acceptable_Use_Policy.doc), elabore uma pequena versão (de 5 a 10 pontos) adequando-a para a realidade de sua organização. Prepare-se para apresentar a sua política no final dessa atividade.

Atividade 6.2 – Antes e durante a instalação

O servidor web Apache deve estar instalado no sistema. Verifique se foi instalado o servidor web Apache na máquina virtual Linux disponibilizada no laboratório. Em caso afirmativo:

1. Interrompa o serviço em execução;
2. Remova o serviço Apache da inicialização com *rcconf*;
3. Localize o nome do pacote de instalação do servidor web Apache e remova esse aplicativo;
4. Identifique três aplicativos que dependem do compilador Perl para funcionar.

Atividade 6.3 – Fortalecendo a segurança

Para cada tópico abaixo, implemente e crie alguma forma de demonstrar a importância dessa configuração para a segurança do ambiente:

1. Atualize o Sistema Operacional;
2. Configure a senha do Grub:

- Crie a senha criptografada com o comando: # grub-mkpasswd-pbkdf2
  - Será solicitada senha, entre com a senha “rnpesr”;
  - Será gerado um hash no formato (grub.pbkdf2.sha512.10000.XXXXXXXXX...);
- Copie e cole o hash montando a estrutura parecida com a que está abaixo no arquivo */etc/grub.d/00_header*.

```
cat << EOF
set superusers=”usuario”
password_pbkdf2 usuario grub.pbkdf2.sha512.10000.XXXXXXXXX...
EOF
```

- Salve o arquivo e saia;
- Execute o comando *# update-grub*;
- Reinicie e teste a edição do Grub durante o boot do sistema.

3. Restrinja o uso do console para o usuário root (dica: lembre-se de criar um usuário, senão você não vai mais conseguir se autenticar no sistema);
4. Desative a reinicialização do sistema com as teclas “CTRL+ALT+DEL”;
5. Desative, na partição */tmp*, a opção de executar arquivos a partir dessa pasta;

6. Implemente, para o usuário “alunoesr”, uma quota de disco na pasta */home* (instale o *winscp* e utilize algum arquivo com mais de 5 Mb). O sistema de quotas deve ser instalado:

```
#apt-get install quota
```

Atividade 6.4 – Fornecendo acesso seguro ao usuário

Utilizando como referência o manual da RNP, implemente e crie alguma forma de demonstrar a importância das configurações do PAM para a segurança do ambiente:

1. Ative o suporte MD5 nas aplicações de login e SSH;
2. Garanta que o usuário root autentique apenas em terminais locais;
3. Crie o grupo wheel.

Atividade 6.5 – Utilizando sudo

Instale e configure o sudo para permitir que o usuário “alunoesr” possa executar o aplicativo passwd como se fosse o root e, dessa forma, trocar a senha de qualquer usuário do sistema (pesquise no Google uma forma de bloqueio caso o usuário “alunoesr” tente trocar a senha do usuário root).

Atividade 6.6 – Calculando e testando umask

Utilizando o manual da RNP como referência, modifique o umask dos usuários para que novos arquivos sejam criados com as seguintes permissões:

1. Leitura e escrita para o dono, leitura e escrita para o grupo e leitura para o resto;
2. Leitura para o dono, sem acesso para o grupo e sem acesso para o resto.

Atividade 6.7 – Limitando o acesso a outras informações de usuários

Utilizando o manual da RNP como referência, modifique o arquivo de configuração */etc/adduser.conf* evitando que, ao criar novos usuários, estes possam acessar o diretório de outros usuários. Demonstre que essa configuração está funcionando perfeitamente.

Atividade 6.8 – Restringindo o acesso de usuários

Crie um usuário chamado “fulano” e não permita que esse usuário possa se autenticar no sistema ou acessar um shell válido. Demonstre que essa configuração está funcionando perfeitamente.

Atividade 6.9 – Implementando o logout automático de usuários ociosos

Utilizando como referência o manual do curso, implemente algum mecanismo que desconecte automaticamente o usuário caso ele fique mais de 1 minuto sem digitar um comando no console.

Atividade 6.10 – Utilizando o chroot

Utilizando o aplicativo chroot, implemente um ambiente onde um usuário só poderá executar o comando *ls* e os comandos básicos disponíveis no shell bash.

Atividade 6.11 – Utilizando o portsentry

Instale e configure o portsentry para identificar possíveis tentativas de invasão. Para testar o funcionamento, remova o comentário da linha PORT_BANNER no arquivo de configuração */etc/portsentry/portsentry.conf* e tente se conectar na porta 11 utilizando a máquina Windows. Acesse o log e verifique se as tentativas estão sendo logadas pelo portsentry.

Atividade 6.12 – Utilizando o tripwire

Instale e configure o tripwire em sua máquina Linux. Crie um breve roteiro que mostre seu funcionamento caso um novo usuário seja adicionado ao sistema.

7

Segurança – Firewall

objetivos

Entender o funcionamento de um firewall, seus tipos e conceitos associados; saber como instalar, configurar e testar um firewall em um ambiente Linux.

conceitos

Firewall para host e para rede, stateless e stateful; netfilter (regras, estados, chains e tabelas); configuração do firewall (políticas, módulos e comandos do iptables).

Firewall

- Filtro de pacotes.
- Atua principalmente nos níveis 3 e 4 do modelo ISO/OSI.
- É uma importante ferramenta de segurança, mas não deve ser a única.

Um firewall, ou filtro de pacotes, é um recurso utilizado para proteger uma máquina ou uma rede através do controle e filtragem dos pacotes/datagramas que entram ou que saem.

Um firewall bem configurado é capaz de controlar com precisão o que entra e o que sai de uma rede e/ou host. No entanto, mesmo bem configurado, um firewall não é garantia de segurança total. Por isso o administrador de redes consciente deve considerar o firewall como uma ferramenta a mais para tornar sua rede segura, e não como a única ferramenta.

Tradicionalmente, um firewall atua nas camadas 3 e 4 do modelo ISO/OSI. Alguns são capazes de também atuar na camada de aplicação, mas são menos comuns devido à sua complexidade. Neste Capítulo, trataremos apenas do primeiro tipo de firewall.

Firewalls podem:

- Proteger apenas um host.
- Implementados na própria máquina.
- Não exigem mudanças na topologia da rede.
- Proteger uma ou mais redes.
- Atuam como roteador.
- Exigem mudanças na topologia da rede: o tráfego da rede deve passar por ele.

Firewalls para host e para rede

Um firewall pode ser utilizado para proteger uma única máquina (host) ou uma rede inteira.

No primeiro caso, o firewall apenas examina os pacotes com origem ou destino ao host em questão. Sua configuração é, em geral, mais simples e não há necessidade de nenhuma modificação na topologia da rede. Já o firewall para uma rede exige que passem através dele todos os pacotes que entram ou saem da rede que está protegendo. Para isso, são necessárias alterações na topologia da rede para acomodar o firewall, que também poderá funcionar como um roteador para essa rede.

Tipos de firewalls

Stateless:

- Cada pacote é analisado sem levar em conta outros pacotes.
- Não existe "conexão" (controle sobre o fluxo).

Stateful:

- Guardam atributos de conexões.
- Reconhecem "estados".
- No Linux: kernel > 2.4.x.

Os firewalls podem ser classificados em duas grandes categorias: stateless e stateful.

Stateless

Firewalls que analisam cada pacote de maneira isolada, levando em conta apenas atributos dos cabeçalhos, como, por exemplo, endereços e portas de origem e destino. Esse tipo de firewall não tem como reconhecer se um determinado pacote está tentando estabelecer uma nova conexão ou se faz parte de uma conexão já estabelecida.

Além da necessidade de se criar duas regras, um atacante malicioso poderia acessar hosts atrás do firewall simplesmente utilizando conexões originadas na porta 80.

No Linux, kernels até a versão 2.2.x implementavam apenas firewalls do tipo stateless.
A partir da versão 2.4.x do kernel, o Linux também passa a suportar firewalls do tipo stateful.

A criação de regras para firewalls desse tipo é mais complexa e menos precisa, porque o administrador precisa criar pares de regras para cada situação. Por exemplo, supondo que o administrador precise permitir que hosts de sua rede acessem páginas web na internet, seria necessário:

- Criar uma regra que permitisse pacotes com endereços de sua rede e portas de origem quaisquer (>1023), e destino qualquer e porta de destino 80.
- Criar uma regra com origem qualquer e porta de origem 80, com destino de sua rede e porta de destino qualquer (>1023).

Stateful

Firewalls capazes de reconhecer e armazenar estados de conexões que passam por ele, como, por exemplo, fluxos TCP. Um firewall desse tipo pode decidir o que fazer com um pacote, baseado no estado desse pacote em relação a uma conexão.

Utilizando o exemplo anterior, o administrador teria de criar apenas a primeira regra e uma segunda regra genérica que permita a passagem dos pacotes que façam parte das conexões estabelecidas.

Netfilter

Nome do código do firewall no Linux. Conceitos:

- Regras.
- Estados.
- Chains.
- Tabelas.

Netfilter é o nome do código presente no kernel do Linux (>2.4.x), que implementa as funções de firewall. Para entender melhor como o netfilter funciona, será necessário estudar alguns conceitos utilizados por ele.

Regras

Possui um padrão de casamento (match) e um alvo (target).

- Padrões de casamento (match):
  - Campos de cabeçalho.
  - Estados de conexões.
- Alvos (target):
  - Aceitar, rejeitar, ignorar, modificar, logar, chains.

Basicamente, uma regra é uma descrição do que fazer com que tipo de pacote. Cada regra possui um padrão de procura e uma ação, também chamada de alvo. Na criação de um padrão, utiliza-se praticamente qualquer campo dos cabeçalhos dos protocolos IP, ICMP, TCP e UDP. Além disso, dentro de uma conexão nesses padrões, é possível utilizar também os estados de um pacote.

Cada tipo de regra aceita um conjunto de ações possíveis. Pode-se aceitar, rejeitar, ignorar, modificar, registrar, marcar pacotes ou até mesmo fazer com que os pacotes sejam avaliados por outros conjuntos de regras. Veremos mais adiante algumas ações possíveis e como construir padrões para serem utilizados nas regras.

Estados

Relação do pacote com um fluxo.

Netfilter reconhece os seguintes estados:

- NEW.
- ESTABLISHED.
- RELATED.
- INVALID.

Como foi dito anteriormente, o netfilter é capaz de reconhecer o estado de uma conexão. Além de facilitar a configuração de regras, isso permite maior controle sobre o que passa pelo firewall. O netfilter pode atribuir quatro estados para um pacote:

- **NEW**: pacotes iniciais de um fluxo;
- **ESTABLISHED**: quando a resposta ao pacote inicial (NEW) chega ao firewall, os próximos pacotes desse fluxo passam a ser identificados como ESTABLISHED;
- **RELATED**: pacotes que de alguma maneira se relacionam a uma conexão já estabelecida. Ao se fazer uma conexão FTP, por exemplo, são abertas duas conexões: uma em que tra-

fegam dados; e outra em que trafegam informações ou controle. Essa segunda conexão poderia ser vista pelo netfilter como sendo RELATED;

- **INVALID**: pacotes que, por alguma razão, não foram identificados.

É importante ressaltar que mesmo protocolos não orientados à conexão, como UDP e ICMP, podem ser tratados pelo firewall de modo similar aos pacotes de protocolos com conexão. Vejamos o caso do *ping*, por exemplo. Quando o firewall recebe um pacote ICMP echo request com um determinado IP de origem ou destino, esse pacote é marcado como NEW. Ao receber o pacote ICMP echo reply correspondente, os pacotes passam a ser classificados como ESTABLISHED.

O kernel armazena os estados das conexões em memória, e permite que essas conexões sejam visualizadas em um arquivo especial dentro do diretório */proc*. Esse arquivo é o */proc/net/ip_conntrack*, que lista, para cada conexão, o protocolo; o número do protocolo; o tempo até a conexão expirar (caso fique inativa); o estado da conexão; e, em seguida, a origem/destino/portas da conexão nos dois sentidos (isto é, primeiro a origem/destino e em seguida o destino/origem).

Chains

- Conjuntos (cadeias) de regras.
- Chains do sistema x Chains do usuário.
- Chains do sistema estão ligadas a pontos especiais que os pacotes percorrem no kernel.
- Regras nas chains do sistema podem ter como alvo as chains do usuário.

Uma chain é uma estrutura que contém regras. O netfilter possui dois tipos de chains: as chains do sistema e as chains criadas pelo usuário.

- **Chains do sistema**: denominadas PREROUTING, INPUT, FORWARD, OUTPUT e POSTROUTING, estão ligadas a pontos especiais no caminho que os pacotes percorrem ao entrar e sair da máquina.
- **Chain de usuário**: só será percorrida se alguma chain do kernel encaminhar pacotes para ela. Ao contrário das chains do kernel, as chains de usuário não fazem parte do caminho que os pacotes percorrem no kernel.

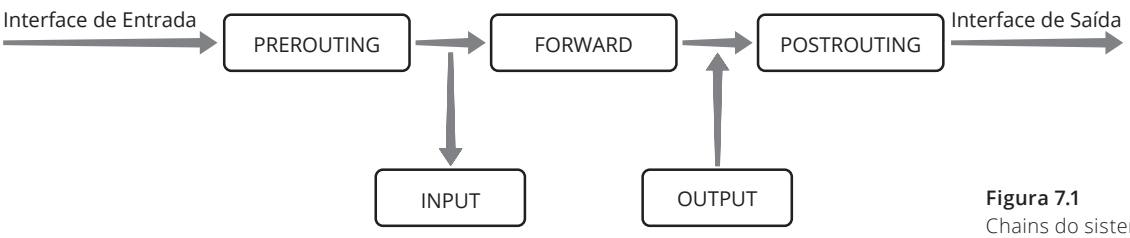


Figura 7.1
Chains do sistema.

A figura mostra como as chains estão organizadas no kernel. Um pacote pode percorrer três caminhos dentro do netfilter:

- Pacotes que venham da rede com destino à máquina percorrem as chains PREROUTING e, em seguida, a chain INPUT.
- Pacotes que tenham origem na máquina percorrem a chain OUTPUT e, em seguida, a chain POSTROUTING.
- Pacotes que venham de uma rede com destino a outra rede (o firewall está atuando como

um roteador) percorrem as chains PREROUTING, FORWARD e, em seguida, a POSTROUTING.

Quando um pacote entra em uma chain, cada regra dessa chain é sequencialmente avaliada em duas situações:

- Até que o pacote combine com uma regra que contenha um alvo do tipo ACCEPT, DROP ou REJECT;
- Até que o pacote atinja o final da chain.

Nesse segundo caso, quando um pacote atinge o final de uma chain sem que tenha casado com alguma regra, é aplicada, então, a política padrão da chain.

A chain PREROUTING recebe esse nome porque ela é avaliada antes que o kernel tome as decisões de roteamento. A chain POSTROUNTING é percorrida após ser feita a decisão de roteamento e logo antes dos pacotes deixarem o netfilter. Essas duas chains são principalmente utilizadas por regras que de alguma maneira alteram o cabeçalho dos pacotes. As chains INPUT, FORWARD e OUTPUT são utilizadas principalmente para as regras que filtram pacotes.

Tabelas

- Estruturas onde são armazenadas as chains.
- Possuem funções específicas.
- Há três tipos de tabelas:
  - Filter: filtra pacotes (observação: único tipo a ser estudado aqui).
  - Mangle: altera e "marca" pacotes.
  - Nat: traduz endereços de rede.

Tabelas são as estruturas onde as chains são armazenadas. O Linux possui apenas três tabelas, e as chains dessas tabelas contêm regras com funções específicas:

- **Filter**: tabela em que são colocadas as regras que filtram pacotes. Essa tabela possui as chains INPUT, OUTPUT e FORWARD.
- **Mangle**: utilizada por regras que alteram ou marcam pacotes. Possui as cinco chains: INPUT, OUTPUT, FORWARD, PREROUTING e POSTROUTING.
- **Nat**: utilizada para as regras de tradução de endereços de rede. Possui as chains PREROUTING, OUTPUT e POSTROUTING.

Configuração do firewall

O que filtrar? Duas abordagens:

- Política padrão DROP e regras específicas para os serviços permitidos.
- Política padrão ACCEPT e regras para bloquear serviços específicos.

Em geral, a configuração de um firewall envolve três passos:

- Decidir o que vai ser filtrado;
- Carregar os módulos do kernel necessários para as tarefas a serem feitas;
- Criar as regras e os scripts para carregá-las no kernel.

A seguir, veremos cada um desses passos.

Políticas de firewall

A decisão sobre o que filtrar vai depender da rede ou máquina que se deseja proteger. Cada cenário exige uma configuração de firewall própria. No entanto, podemos ter duas abordagens básicas ao criar um firewall.

Alguns firewalls são criados com uma política padrão de DROP. O firewall descarta todos

os pacotes para os quais não existem regras específicas que os aceitem. Em geral esses firewalls são mais restritivos na filtragem de pacotes. Outros firewalls seguem o princípio oposto: deixam passar qualquer pacote, e as regras são criadas para bloqueio de determinados tipos de pacote.

A escolha por uma ou outra abordagem vai depender do que se deseja proteger e da política de segurança adotada pela instituição. Em geral os firewalls que protegem intranets são do primeiro tipo, enquanto que o segundo tipo é utilizado em roteadores de borda para conter determinados ataques e/ou worms.

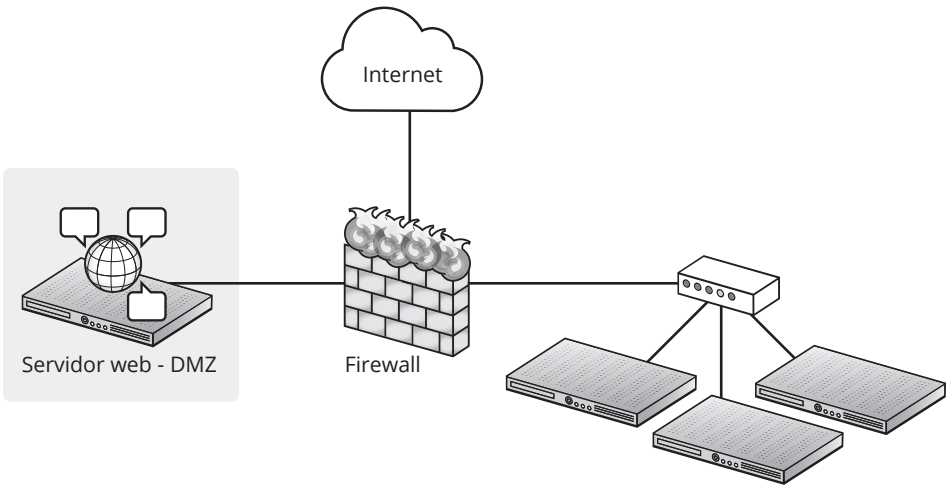


Figura 7.2
DMZ (Zona Desmilitarizada).

Em algumas situações, também é desejável oferecer alguns serviços para a internet e, ao mesmo tempo, proteger a intranet. Nesses casos é comum a criação das chamadas zonas desmilitarizadas ou DMZ. Uma DMZ é uma sub-rede, na qual são colocadas as máquinas que oferecem serviços para a internet, como servidores web e de e-mail. Ao se configurar um firewall com uma DMZ, o administrador cria regras específicas de acesso para a DMZ e para a intranet.

Módulos do kernel

Principais módulos:

- ip_tables
- iptables_filter
- iptables_mangle
- iptables_nat
- ip_conntrack
- ip_conntrack_ftp
- ipt_state
- ipt_LOG
- ipt_REJECT

Para criar um firewall no Linux é necessário que o kernel tenha suporte ao netfilter. O netfilter tanto pode ser compilado embutido (built-in) no kernel, como em forma de módulos.

Se o netfilter estiver na forma de módulos, pode ser necessário carregá-los com o comando *modprobe* antes de inserir as regras no kernel. Alguns kernels têm suporte ao carregamento automático de módulos.

O kernel que acompanha a maioria das distribuições traz o netfilter compilado na forma de módulos. Alguns administradores argumentam que o netfilter compilado *built-in* tem uma performance ligeiramente melhor e é mais seguro, já que é possível compilar o kernel sem suporte a módulos. Isso evita que um invasor carregue módulos maliciosos no kernel.

Outros administradores argumentam que um kernel modular permite que somente os módulos necessários sejam carregados, o que evita desperdício de memória, além de permitir que um mesmo kernel possa ser utilizado em máquinas com funções diferentes (servidor, firewall, desktop etc.).

Principais módulos do netfilter e suas funções:

- **ip_tables**: habilita o suporte ao netfilter;
- **iptables_filter**: habilita o suporte à tabela filter;
- **iptables_mangle**: habilita o suporte à tabela mangle;
- **iptables_nat**: habilita o suporte à tabela NAT;
- **ip_conntrack**: habilita o suporte ao reconhecimento de conexões;
- **ip_conntrack_ftp**: habilita o suporte ao reconhecimento das conexões do FTP ativo (conexões RELATED);
- **ipt_state**: habilita a permissão para regras baseadas no estado da conexão;
- **ipt_LOG**: habilita o suporte ao alvo LOG (veja as tabelas de alvos no item seguinte);
- **ipt_REJECT**: habilita o suporte ao alvo REJECT.

Manipulação de regras

- Feita com o comando *iptables*.
- Cada tipo de regra admite opções diferentes.
- Sintaxe para criação ou remoção de regras:

```
# iptables [-t TABLE] -[AID] CHAIN [N] MATCH -j TARGET
```

- Comando iptables

O comando *iptables* é utilizado para a manipulação de regras e chains dentro do kernel. Esse comando admite uma grande variedade de opções. A seguir as mais comuns:

- Manipulação de regras;
- Criar e remover regras.

Opções para criação e remoção de regras:

- -A – “Append”, adiciona regras ao final da chain.
- -I – “Insert”, insere a regra na posição N; se não for indicada a posição, insere no começo da chain.
- -D – “Delete”, remove uma regra. Há dois modos:
  - Indicando o número da regra.
  - Usando as mesmas opções (padrão e alvo) com as quais a regra foi criada.
- -t – admite as tabelas filter, mangle e nat. Caso a tabela não seja definida, o iptables considera que a escolhida é a filter.

No slide são mostradas as opções utilizadas para criar e remover regras do firewall iptables. É importante destacar que a opção *–t* deve ser informada sempre que a tabela a ser utilizada for diferente da filter.

Padrões de casamento

- -s ou --src <IP>[/bits] – casa com a origem do pacote.
- -d ou --dst <IP>[/bits] – casa com o destino do pacote.
- -i ou --in-interface <interface> – interface de entrada.
- -o ou --out-interface <interface> – interface de saída.
- Dica: use "!" para negar uma opção:
  iptables -A INPUT -i eth0 ! -s 192.168.0.1 –j\ ALVO

São muitas as opções que definem o padrão de casamento (MATCH) de uma regra, como pode ser verificado a seguir (não estão listadas todas as opções):

Opção	Opção (longa)	Valor	Significado
-s	--src	<IP>[/bits]	Casa com o endereço de origem do pacote.
-d	--dst	<IP>[/bits]	Casa com o endereço de destino do pacote.
-i	-in-interface	<interface>	Casa com a interface pela qual o pacote chegou.
-o	--out-interface	<interface>	Casa com a interface pela qual o pacote vai sair.
-p	--protocol	tcp,udp,icmp	Casa com o protocolo do pacote ou datagrama.
--sport	--source-port	<porta>[:porta]	Casa com a porta (ou intervalo) de origem do pacote; precisa ser usado com a opção -p (tcp/udp).
--dport	--destination-port	<porta>[:porta]	Casa com a porta (ou intervalo) de destino do pacote; necessita ser usado com a opção -p (tcp/udp).
--icmp	-type	<tipo/código>	Casa com o tipo/código de um pacote icmp.
-m	- -match	<módulo>	Habilita o modo de opções entendido.

Uma opção pode ser negada se a precedermos com um "!". Assim, "! -s 192.168.0.1/32" casa com qualquer origem, menos com 192.168.0.1/32.

- p ou –protocol <tcp,udp,icmp> – protocolo do pacote.
- --sport ou --source-port <porta>[:porta] – porta(s) de origem do datagrama.
- --dport ou --destination-port <porta>[:porta] – porta(s) de destino do datagrama.

Exemplo:

```
iptables -A FORWARD -p tcp --dst 192.168.0.1 --dport 80 -j ALVO
```

- -m ou --match <módulo> – habilita novas opções.

Exemplo:

- -m state – carrega o módulo de estados.
- -m state --state <NEW, ESTABLISHED, RELATED,INVALID> – casa com o estado da conexão.

Além das opções da tabela anterior, outras mais podem ser usadas ao se especificar a

opção --match:

- **--limit (valor X/time)**: limita a média de matchs (o que exceder esse número é descartado). Time pode ser /second, /minute, /hour ou /day. Essa opção e a seguinte são, em geral, utilizadas com o alvo LOG, para limitar o número de pacotes logados.
- **--limit-burst (valor X):** número máximo inicial de matchs. Esse valor é recarregado em 1 a cada intervalo de tempo em que o limite não foi alcançado, até chegar ao valor especificado. Utilize essa opção para permitir rajadas.
- **--state (valores NEW, ESTABLISHED, RELATED, INVALID)**: casa com o estado de uma conexão (utilize com a opção -m state).
- **--tcp-flags (valor MASK FLAGS)**: examina as flags de um pacote TCP. MASK define o que deve ser examinado e FLAGS define quais bits devem estar ligados. Exemplo: SYN, ACK ACK = examina SYN e ACK, e casa com pacotes que tenham SYN desligado e ACK ligado.
- **--tos (valor *name*)**: casa com o campo "type of service". Para ver os nomes e valores, use o comando *iptables -m tos –h*.
- --**ttl**: casa com o valor do campo TTL do pacote.
- --**syn**: casa com pacotes do tipo SYN do protocolo TCP.
- --tcp-flags <mask, flags> – examina as flags definidas em "mask" e casa com flags ligadas (bit 1).
  - Exemplo: iptables -A FORWARD -p tcp --tcp-flags SYN,ACK SYN -j ALVO
    - Examina as flags SYN e ACK. A flag SYN deve estar "ligada".

Definição de alvo de uma regra (exemplo tabela filter).

- ACCEPT: aceita um pacote.
- REJECT: rejeita um pacote e, opcionalmente, envia um ICMP tipo <type> para a origem do pacote.
- DROP: descarta silenciosamente os pacotes.
- LOG: registrar o pacote.
- RETURN: reenvia o pacote para a sua chain de origem.

Por fim, devemos especificar o alvo de uma regra. Como cada tabela possui alvos diferentes e com opções diferentes, veremos apenas os alvos da tabela filter:

- **ACCEPT**: aceita um pacote.
- **REJECT --reject-with <type>**: rejeita um pacote e, opcionalmente, envia um ICMP tipo <type> para a origem do pacote.
- **DROP**: descarta silenciosamente os pacotes.
- **LOG --log-level X –log-prefix Y**: loga os headers de um pacote e, opcionalmente, utiliza o priority X (veja syslogd.conf) e o prefixo Y, que é uma string de até 29 letras utilizada para ajudar a identificar o pacote de regra.
- **<chain>**: envia o pacote para uma chain do usuário.
- **RETURN**: reenvia o pacote para a sua chain de origem.

Os dois últimos alvos (<chain> e RETURN) merecem uma explicação adicional. Suponha que foi criada uma chain para tratar apenas de pacotes ICMP, e que essa regra foi chamada de ICMP_FILTER. É possível ter, então, uma regra na chain FORWARD do tipo:

```
# iptables -A FORWARD -p icmp -j ICMP_FILTER
```

Pacotes que casem com a regra anterior seriam jogados na chain ICMP_FILTER. O alvo RETURN é usado para fazer com que um pacote retorne à chain de onde foi encaminhado. Por exemplo, para que os pacotes com origem em 192.168.1.1 retornem à chain FORWARD, basta criar a regra:

```
# iptables -A ICMP_FILTER -s 192.168.1.1 -j RETURN
```

Outros exemplos de regras:

- "Dropa" (DROP) tudo que chega ao firewall com destino ao host 192.168.0.3:

```
# iptables -A FORWARD -d 192.168.0.3/32 -j DROP
```

- Remove a regra anterior:

```
# iptables -D FORWARD -d 192.168.0.3/32 -j DROP
```

- Bloqueia a porta 23 da máquina local para acessos vindos pela interface eth0:

```
# iptables -A INPUT -i eth0 -p tcp --dport 23 -j REJECT
```

- "Loga" (LOG) tentativas de acesso à porta 161 UDP vindas de fora da rede 192.168.0.0/24, limitando o número de match para não encher rapidamente o log:

```
# iptables -A FORWARD -p udp --dport 161 ! -s 192.168.0.0/24 -j LOG
--limit 1/second
```

Observe que uma regra de LOG não é terminal, ou seja, o pacote continua a percorrer as regras da chain. As regras a seguir logam pacotes TCP SYN com destino à porta 445, acrescendo o prefixo SCAN à entrada do log e, em seguida, descartam o pacote:

```
# iptables -A FORWARD -p tcp --dport 445 --syn -j LOG -log- prefix
"SCAN"

# iptables -A FORWARD -p tcp --dport 445 --syn -j DROP
```

Listagem de regras

Formato:

```
 # iptables -L [ CHAIN ] [ -t table]
```

Exemplos:

- # iptables –L
- # iptables -L -t mangle
- # iptables -L FORWARD -t filter

Para listar o conteúdo de uma chain, basta utilizar o comando:

```
iptables -L [ CHAIN ] [ -t table]
```

Lembrando que quando não é especificada nenhuma tabela, o padrão é mostrar a tabela filter. Exemplos:

- Mostra todas as chains da tabela filter:

```
# iptables -L
```

- Mostra todas as chains da tabela mangle:

```
# iptables -L -t mangle
```

- Mostra a chain FORWARD da tabela filter:

```
# iptables -L FORWARD -t filter
```

Duas opções muito úteis ao se listar chains são *-v* e *-x*. Elas permitem saber quantos pacotes e quantos bytes percorreram cada chain/regra.

Manipulação de chains

Ajuste da política padrão de uma chain (o que fazer caso nenhuma regra se aplique ao pacote):

```
# iptables -t filter -P FORWARD DROP
```

Criar e remover chains:

```
# iptables -t filter -N servicos_tcp
```

Utilizar chains de usuário:

```
# iptables -A INPUT -p tcp --dport 80 -j servicos_tcp
```

Limpar chains:

```
# iptables -t table -F <chain>
```

É possível utilizar apenas as chains do sistema para construir um firewall. Porém, nem sempre isso é desejável. Isso porque, à medida que cresce o número de regras, o desempenho pode diminuir (um pacote precisa ser avaliado por um número maior de regras), e pode aumentar a complexidade de sua manutenção.

Para reduzir esses problemas, é possível criar chains novas. Veja as vantagens nesse exemplo: imagine um firewall que possua 50 regras para pacotes TCP, 10 regras para pacotes UDP, e mais 10 regras para pacotes ICMP. Se todas essas regras fossem aplicadas em uma única chain, um pacote que atravessasse essa chain poderia ser avaliado por até 70 regras. Para evitar isso, poderiam ser criadas chains específicas para tratar de cada protocolo, e, opcionalmente, dividir em mais de uma chain o tratamento das regras TCP. Isso diminuiria as regras pelas quais um pacote seria avaliado, além de tornar mais fácil a leitura e manutenção do firewall.

Ajuste da política padrão de uma chain

As chains de sistema possuem políticas padrão, isto é, especificações das ações que devem ser tomadas caso nenhuma regra da chain case com o pacote. O valor padrão para as chains do sistema é ACCEPT, o que faz com que qualquer pacote que não tenha sido negado ou rejeitado seja automaticamente aceito. Esse valor pode ser alterado para DROP ou REJECT com a opção *-P <chain> <política>* do comando *iptables*. No exemplo seguinte, a política padrão da chain FORWARD da tabela filter será alterada para DROP:

```
# iptables -t filter -P FORWARD DROP
```

Normalmente, a política padrão altera o modo como serão escritas as regras. Em uma chain com política padrão DROP, em geral são escritas regras com alvos ACCEPT; e para chains com política ACCEPT, regras com alvos DROP ou REJECT.

Criar e remover chains

Para criar uma chain, basta utilizar o comando *iptables* com a opção *-N <nome>*. É necessário também informar em qual tabela será criada a chain. Exemplo:

```
# iptables -t filter -N servicos_tcp
```

Para remover uma chain, a opção utilizada é *-X*. Ao utilizar essa opção, é possível indicar a chain que desejamos remover. Se utilizarmos apenas *–X*, serão removidas todas as chains de usuário. Exemplos:

```
# iptables -t filter -X servicos_tcp

# iptables -X
```

Utilizar chains de usuário

Como já foi citado, para que um pacote atravesse uma chain de usuário é necessário que em alguma chain do sistema haja uma regra que tenha como alvo uma chain de usuário. Exemplo:

```
# iptables -A INPUT -p tcp --dport 80 -j servicos_tcp
```

As regras dessa chain poderão aceitar, descartar ou rejeitar um pacote como as outras regras. Ao chegar ao fim da chain sem que haja um casamento com alguma regra, o pacote retornará à chain de origem. Também é possível utilizar o alvo RETURN com esse objetivo. Exemplo:

```
# iptables -A servicos_tcp -s 192.168.0.1 -j RETURN
```

Limpar chains

Já foi visto como remover uma regra de uma chain. Mas muitas vezes pode ser necessário remover todas as regras de uma chain. Em vez de remover regra por regra, pode-se utilizar a opção *-F* para limpar (flush) todas as regras de uma só vez. Exemplo:

```
# iptables -t table -F <chain>
```

Comandos muito utilizados quando precisamos manipular um grande conjunto de regras:

- iptables-save:
  - Lê todas as regras no kernel e exibe na saída padrão do shell.
  - Exemplo:

```
# iptables-save > regras_firewall
```

- Iptables-restore:
  - Executa a operação inversa, ou seja, recebe a entrada padrão do shell e insere no kernel.
  - Exemplo:

```
# iptables-restore <regras_firewall
```

Embora bastante poderoso, o comando *iptables* apresenta algumas desvantagens ao lidar com um grande número de regras. Para cada regra removida ou inserida, o comando *iptables* precisa consultar todas as regras já existentes na chain que ele está manipulando.

E quanto mais regras há, mais lenta se torna essa operação. Para resolver esse problema é possível utilizar os comandos *iptables-save* e *iptables-restore*, capazes de salvar e inserir regras com mais eficiência.

O comando *iptables-save* lê todas as regras armazenadas no kernel e as exibe na saída padrão do shell em um formato similar ao utilizado pelo comando *iptables*, próprio para a

inserção das regras em bloco. Para salvar as regras em um arquivo, basta direcionar a saída do comando para um arquivo. Exemplo:

```
# iptables-save > regras_firewall
```

O comando *iptables-restore* executa a operação inversa, ou seja, recebe pela entrada padrão do shell um conjunto de regras no formato do *iptables-save* e as insere no kernel. Para utilizar o arquivo do exemplo anterior como entrada padrão, basta utilizar o comando:

```
# iptables-restore <regras_firewall
```

Outras opções:

- Habilitando o repasse de pacotes.
- Ajuste do número de conexões registradas.

Dicas:

- Liberar a interface de loopback.
- Pacotes que atravessam a máquina devem ser filtrados na chain FORWARD, e pacotes que tenham origem ou destino na própria máquina devem ser filtrados nas chains OUTPUT e INPUT.
- As regras de uma chain são avaliadas em sequência.
- Sempre que possível, insira antes na chain as regras que casarão com mais pacotes.

Além da criação das regras e chains, muitas vezes é necessário ajustar outros parâmetros do kernel e do netfilter para obter o resultado desejado para o firewall. Veremos alguns desses parâmetros a seguir.

Habilitando o repasse de pacotes

Por padrão, o Linux não faz o repasse de pacotes entre suas interfaces, ou seja, ele não atua como um roteador. Quando construímos um firewall para uma rede, é necessário que o sistema seja capaz de encaminhar pacotes vindos de uma rede para outra. Para isso, é necessário habilitar essa função, chamada de IP forwarding. Para habilitá-la, é necessário mudar o valor do arquivo */proc/sys/net/ipv4/ip_forward* para 1:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

O Debian permite tornar essa configuração permanente. Basta alterar a seguinte linha do arquivo */etc/sysctl.conf*:

```
net.ipv4.ip_forward = 1
```

Ajuste do número de conexões registradas

O número máximo de conexões que o kernel pode acompanhar (connection tracking) é definido por padrão, em função da quantidade de memória da máquina. Esse valor fica armazenado no arquivo */proc/sys/net/ipv4/ip_conntrack_max*. Caso o firewall seja colocado em uma rede com muitos hosts/conexões, esse valor pode não ser suficiente. Podemos alterar esse valor da seguinte maneira:

```
# echo 65535 > /proc/sys/net/ipv4/ip_conntrack_max
```

Esse comando pode ser incluído em um script de firewall. Outra forma de alterar esse valor é através da inclusão da linha a seguir no arquivo */etc/sysctl.conf*:

```
net.ipv4.ip_conntrack_max=65535
```

O valor 65535 é o valor máximo para os kernels da série 2.4.x.

Dicas

Ao se escrever regras para um firewall, as seguintes dicas podem ser úteis:

- Como o sistema usa a interface de loopback para a comunicação de alguns processos, pode ser necessário liberar o tráfego de pacotes nessa interface.
- Pacotes que atravessam a máquina devem ser filtrados na chain FORWARD, e pacotes que tenham origem ou destino na própria máquina devem ser filtrados nas chains OUTPUT e INPUT.
- As regras de uma chain são avaliadas em sequência. Logo, a ordem na qual elas são inseridas influencia no resultado final.
- Sempre que possível, insira antes na chain as regras que casarão com mais pacotes. Isso evita que um pacote tenha que percorrer um grande número de regras.

Roteiro de Atividades 7

Atividade 7.1 – Módulo ip_conntrack

Verifique os módulos carregados no kernel. Caso o módulo ip_conntrack não esteja carregado, carregue-o com o comando *modprobe*. Em seguida:

1. Verifique o conteúdo do arquivo */proc/net/ip_conntrack*.
2. Estabeleça uma conexão qualquer com outra máquina. Por exemplo, telnet www.google.com 80. Em outro shell, verifique o estado do arquivo *ip_conntrack*. Como essa conexão é mostrada?
3. Tente estabelecer uma conexão com um IP não utilizado da sua rede. Como essa tentativa de conexão é mostrada?

Atividade 7.2 – Política padrão

Mude a política padrão da chain de OUTPUT para DROP. Em seguida:

1. Tente estabelecer uma conexão http. É possível? Por quê?
2. Crie uma regra para permitir que sua máquina estabeleça conexões http.
3. Mude a política padrão da chain INPUT também para DROP. É ainda possível estabelecer conexões http?
4. Crie uma regra na chain INPUT para tornar as conexões http possíveis. Não utilize regras com estado.

Atividade 7.3 – Firewall Stateful

Continuando a atividade anterior:

1. Remova as regras da chain INPUT.
2. Crie uma regra genérica para permitir conexões já estabelecidas (utilizando estados). Verifique se é possível estabelecer conexões http.
3. Remova todas as regras criadas e altere as políticas padrão para ACCEPT.

Atividade 7.4 – Firewall de host

Crie um pequeno firewall para a sua máquina com as seguintes características:

1. Tráfego liberado através da interface de loopback.
2. Permissão a qualquer conexão feita a partir da sua máquina.
3. Bloqueio (DROP) a conexões externas com destino à sua máquina.

Atividade 7.5 – Liberando serviços no firewall

Altere o firewall da atividade anterior, para permitir que o serviço SSH seja acessado de qualquer máquina. E também para que pacotes ICMP sejam permitidos, se vierem de máquinas da sua rede.

Atividade 7.6 – Criando chain

Ainda utilizando o firewall da Atividade 7.4, crie uma chain chamada log. Para essa chain, crie:

1. Uma regra para logar pacotes TCP com a flag syn ligada, que deve ter o prefixo “TCP”.
2. Uma regra para logar pacotes ICMP, que deve logar com o prefixo “ICMP”.
3. Crie as regras necessárias nas outras chains, para que todos os pacotes percorram a chain log.
4. Teste as regras e verifique se o firewall está logando pacotes TCP e ICMP.

Atividade 7.7 – Firewall de rede

Antes de iniciar esta atividade, remova todas as regras criadas nas atividades anteriores.

Nas atividades anteriores foram feitos firewalls apenas para o host. Nesta atividade e nas próximas será criado um firewall para uma rede. Para tanto, considere que sua máquina possui duas interfaces de rede. Mantenha o endereço ou gateway da interface eth0 da máquina que você está utilizando. Considere que ela é a sua interface de acesso à internet. Imagine que sua máquina possui uma interface eth1 com endereço 192.168.200.1/24, e que essa é a interface de acesso a intranet. Veja a figura a seguir:

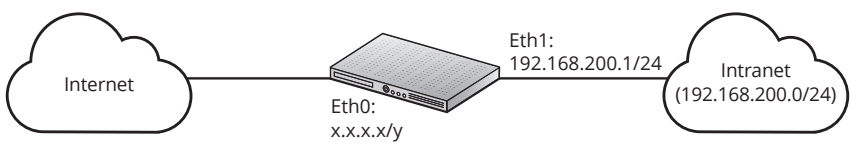


Figura 7.3
Interface de acesso à internet.

Leve em conta o cenário da figura e, nesse momento, apenas o acesso ao firewall. Faça:

1. Regras para liberar o acesso através da interface de loopback;
2. Regras para evitar o IP spoofing (falsificação do IP de origem do pacote) na chain INPUT;
3. Permita que os hosts da intranet possam acessar o serviço SSHD do firewall;
4. Bloqueie qualquer acesso a outros serviços/portas da máquina do firewall.

Atividade 7.8 – Firewall de rede (continuação)

Continuando a atividade anterior:

1. Habilite o IP forwarding;
2. Crie regras para evitar o IP spoofing (use a chain FORWARD);
3. Crie regras que permitam conexões TCP da intranet para a internet. Utilize a política DROP na chain FORWARD;
4. Permita conexões UDP apenas com origem na intranet e com destino para a porta 53 (DNS).

Atividade 7.9 – Worm

Suponha que exista um novo worm na internet. Ele infecta máquinas e, em seguida, tenta infectar outras máquinas na internet através da porta TCP 31337. Crie regras para bloquear as conexões de saída da sua intranet e logue as tentativas de conexão a essa porta com o prefixo WORM.

Atividade 7.10 – DMZ

Para resolver esta atividade, remova todas as regras criadas nas atividades anteriores.

Para esta atividade, considere que o firewall possui três interfaces de rede: eth0, com endereço 10.0.0.1/30; eth1, com endereço 192.168.0.1/24; e eth2, com endereço 192.168.1.1/24. A interface eth0 está conectada ao roteador de borda (internet). A eth1 está conectada na sua intranet, e a eth2 está conectada a uma rede com os seus servidores de web e e-mail (considere-a como a sua DMZ). A figura a seguir resume a topologia da rede.

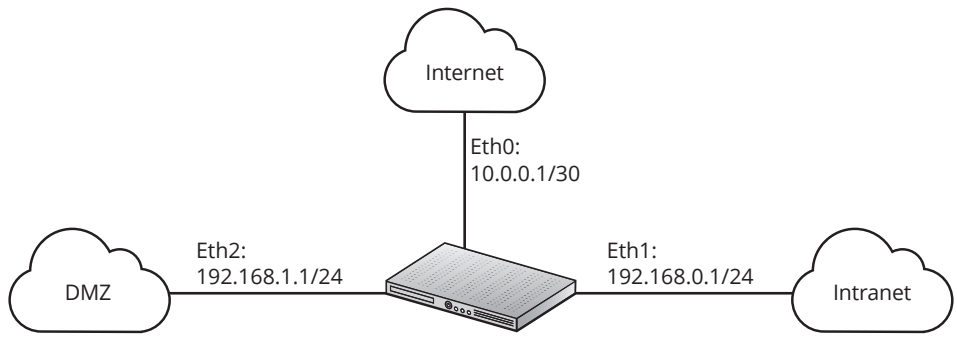


Figura 7.4
Topologia da rede da Atividade 7.10.

Considerando este cenário:

1. Crie um par de chains para a comunicação entre cada dupla de redes (isto é, internet-intranet, internet-DMZ e intranet-DMZ). Cada chain de um par conterá regras para conexões em uma única direção (por exemplo, intranet -> dmz e dmz -> intranet).
2. Crie as regras na chain FORWARD necessárias para encaminhar os pacotes para cada uma das chains criadas no item anterior. Apenas pacotes no estado NEW devem ser encaminhados.
3. Crie uma regra genérica na chain FORWARD, para permitir que sejam aceitos pacotes de conexões estabelecidas ou relacionadas. Mude a política da chain de FORWARD para DROP.

8

Interconexão de redes

objetivos

Entender os conceitos e os tipos de tradução de endereços de rede (NAT); configurar os tipos de NAT em uma rede; aprender o conceito e os tipos de túneis; configurar túneis; entender e aplicar os conceitos de políticas de roteamento e de múltiplas tabelas de roteamento.

conceitos

Tradução de endereços de rede (tipos de NAT, SNAT e DNAT); roteamento avançado através de túneis (comando *ip tunnel*; túneis IPIP, GRE e SIT); tabelas de roteamento (RPDB).

Tradução de endereços de redes

Network Address Translation (NAT):

- Também conhecido como IP-Masquerading.
- Permite alterar campos do cabeçalho IP.
- Frequentemente usado quando não há endereços "públicos" para todos os hosts de uma rede.
- Viabiliza a conexão de toda uma rede à internet com apenas um endereço "público".

Ao tratarmos do tema firewall, vimos como uma máquina Linux pode se transformar em um roteador. Basicamente, ao habilitar o IP Forward é possível transformar o host em um roteador. Neste capítulo serão estudadas algumas funcionalidades mais avançadas de roteamento, como a tradução de endereços de rede ou Network Address Translation (NAT).

NAT é composto de técnicas que consistem na alteração dos endereços IP de origem e/ou destino de pacotes que atravessam um roteador ou firewall. Ele é contrário a alguns dos princípios da internet, como o da conectividade fim-a-fim e o da alteração mínima dos pacotes no caminho entre a origem e destino. No entanto, o NAT tem sido utilizado com bastante frequência em certos cenários. O principal deles é quando não há endereços IP válidos – isto é, únicos e roteados na internet – disponíveis para todos os hosts de uma rede. Utilizando técnicas de NAT, é possível conectar toda uma rede à internet a partir de um único IP válido.

Tipos de NAT

NAT:

- Também chamado de "NAT estático".
- Faz a tradução 1:1 (de um endereço IP para outro).

NAPT:

- Tradução de endereços e portas de rede.
- Usado quando mais de um host precisa utilizar um único endereço IP.
- Também pode ser classificado em dois outros tipos.
- Source NAT (SNAT).
  - Destination NAT (DNAT).

As técnicas de NAT podem ser classificadas em dois tipos básicos:

- **NAPT**: em geral chamado apenas de NAT, se refere à tradução de endereço e portas de rede. É utilizado quando há mais de um host compartilhando um único endereço IP válido.
- **NAT estático**: outra forma de NAT, simplesmente faz a tradução 1:1 de um endereço válido para um inválido, sem haver mapeamento de portas.

O NAPT também pode ser classificado em dois outros tipos:

- **Source NAT (SNAT)**: são alterados o endereço e a porta de origem.
- **Destination NAT (DNAT)**: são alterados o endereço e a porta de destino.

A seguir, será detalhado o funcionamento desses dois tipos de NAT e o modo como eles podem ser configurados no Linux.

Source NAT (SNAT)

Uso frequente para conectar redes/hosts à internet.

Como uma rede com endereço privado acessa a internet?

- Com endereço privado, o pacote geralmente não atinge o destino (filtros).
- Se atingir, é impossível enviar uma resposta de volta.

Source NAT é utilizado, principalmente, para conectar redes ou hosts com endereços IP privados à internet. Isso é feito através da alteração dos endereços IP de origem dos pacotes.

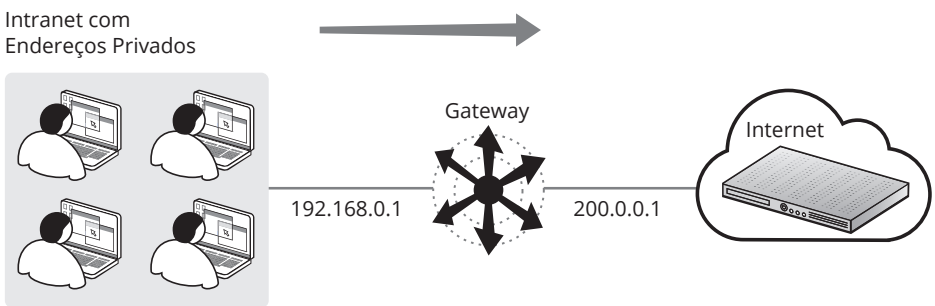


Figura 8.1
Source NAT (SNAT).

Considere o exemplo ilustrado na figura acima. Deseja-se permitir que uma intranet com endereços 192.168.0.0/24 acesse hosts na internet. Para tanto, suponha que exista uma máquina gateway que possui duas interfaces: uma com endereço 192.168.0.1; e outra conectada à internet com o endereço IP público 200.0.0.1.

No exemplo, se o host 192.168.0.5 tentar se conectar a um servidor web da internet sem usar o NAT, ocorrerá o seguinte:

- Os pacotes com destino ao servidor serão rejeitados por algum roteador no meio do caminho, pois é comum roteadores da internet não aceitarem pacotes com origem ou destino de endereços IP privados;
- O servidor web não poderá enviar a resposta por não saber como rotear pacotes com endereços de destino privado.

Em geral, o gateway NAT não altera a porta de origem do pacote. No entanto, se existirem duas conexões com a mesma porta de origem, o gateway NAT terá de alterar a porta de origem de uma dessas conexões.

Em ambos os casos a conexão não será possível. Para evitar essa situação, a solução é usar o SNAT no gateway da rede. Ocorrerá o seguinte:

- Quando os pacotes originários do host 192.168.0.5 chegarem ao gateway NAT, eles terão seu endereço de origem alterado para o endereço público (200.0.0.1 no nosso exemplo), e o gateway passará a armazenar informações sobre essa conexão;
- O servidor web utilizará o endereço público do gateway NAT para estabelecer a conexão;
- Ao chegarem ao gateway, os pacotes terão o endereço de destino alterado para o endereço do host que iniciou a conexão (ou seja, 192.168.0.5).

Configuração do SNAT

- No Linux, também é feita pelo netfilter através do comando *iptables*.
- As regras para SNAT são aplicadas na chain POSTROUTING.

No Linux, o NAT é feito através do netfilter e é configurado com o comando *iptables*.
As regras de NAT são aplicadas nas chains da tabela NAT.

As regras para o SNAT devem ser aplicadas na chain POSTROUTING, que é percorrida após o **kernel** tomar a decisão de roteamento. O alvo para essas regras é o SNAT e ele deve ser utilizado com a opção *--to-source* indicando o endereço IP para o qual os endereços de origem devem ser alterados.

Kernel
Núcleo ou cerne (em inglês: kernel) é o componente central do sistema operativo da maioria dos computadores. Ele serve de ponte entre aplicativos e o processamento real de dados em nível de hardware.

O exemplo a seguir faz o SNAT de todos os pacotes que deixam a máquina através da interface eth1 ao utilizar o IP público 200.0.0.1:

```
# iptables -t nat -A POSTROUTING -o eth1 -j SNAT -- to-source
200.0.0.1
```

A opção *--to-source* aceita intervalos de IP e intervalos de portas. Para tal, a sintaxe utilizada é:

```
--to-source IP[-IP]:[porta-porta]
```

Quando um intervalo de endereços IP é especificado, o kernel alterna os endereços de origem que serão trocados nos pacotes de saída. Quando um intervalo de portas é utilizado, o netfilter o utilizará para as portas de origem dos pacotes de saída.

O netfilter possui um modo especial de SNAT, chamado masquerade, que utiliza o alvo MASQ. Esse modo é utilizado quando a interface de saída não possui um IP fixo. Na utilização do alvo MASQ, o netfilter automaticamente usa o endereço IP da interface de saída. De maneira análoga ao alvo SNAT, é possível especificar o intervalo das portas de saída com a opção *--to-ports <porta-porta>*. O exemplo a seguir faz o SNAT, utilizando a interface ppp0:

```
# iptables -t nat -POSTROUTING -o ppp0 -j MASQ
```

Assim como as outras regras do netfilter, também é possível especificar padrões de casamento mais específicos para as regras de NAT. No exemplo seguinte será feito o SNAT apenas para conexões vindas do host 192.168.0.5 e com destino à porta 80 TCP:

```
# iptables -t NAT -A POSTROUTING -s 192.168.0.5 -p tcp --dport 80 -j
SNAT --to-source 200.0.0.1
```

Exemplos de pacotes que deixam o host com a interface eth1 trocando a origem para 200.0.0.1:

iptables -t nat -A POSTROUTING -o eth1 –j\ SNAT --to-source 200.0.0.1

A opção - -to-source aceita um intervalo de IP e porta:

--to-source IP[-IP]:[porta-porta]

Masquerade é um modo especial de SNAT, que usa o alvo MASQ e, automaticamente, o IP da interface de saída:

iptables -t nat -POSTROUTING -o ppp0 -j\ MASQ [--to-ports porta[-porta]]

Também é possível especificar padrões específicos:

iptables -t NAT -A POSTROUTING -p tcp --\ dport 80 -j SNAT --to-source 200.0.0.1

DNAT manipula endereços e portas de destino. Uso comum:

- Permitir que hosts na internet acessem hosts de uma rede com endereços privados.
- Balanceamento de serviços (balancear requisições com destino à porta 80, por exemplo).

Cuidado: no Linux, o DNAT altera os pacotes em apenas um único sentido (ao contrário do SNAT).

O destination NAT, como o próprio nome indica, manipula os cabeçalhos dos pacotes ou datagramas e altera os seus IPs ou portas de destino. Uma situação comum em seu uso é quando se quer permitir que hosts na internet acessem serviços em hosts dentro de uma intranet com endereços IP privados. Em tal cenário, a conexão entre intranet e internet é feita, em geral, através de um gateway que possui um endereço válido e um inválido. Esse gateway pode encaminhar requisições feitas ao seu endereço IP válido para hosts na intranet.

Outra situação comum de uso do DNAT é no balanceamento de conexões destinadas a um determinado serviço. Por exemplo, um gateway DNAT poderia receber requisições http (porta 80) e balancear essas conexões entre um conjunto de servidores web.

Um cuidado adicional deve ser tomado ao se utilizar o DNAT. No Linux, o DNAT altera apenas os endereços (e, opcionalmente, portas) de destino dos pacotes em um único sentido da conexão. Já o SNAT altera os pacotes nos dois sentidos da conexão. Logo, é comum a utilização do DNAT associado com o SNAT.

Configuração do DNAT

A configuração do DNAT é análoga ao SNAT:

- As regras devem ser aplicadas na tabela PREROUTING.
- Exemplo de uso para trocar o destino de pacotes que chegam da interface eth0, com o destino de 200.0.0.1 para 192.168.0.1

  # iptables -t nat -A PREROUTING -i eth0 -d \ 200.0.0.1 -j DNAT --to-destination 192.168.0.1
- Exemplo de balanceamento da porta 80 para conexões vindas pela interface ppp0, para os servidores de 192.168.0.3 a 192.168.0.5:

  # iptables -t nat -A PREROUTING -i ppp0 –p \ tcp --dport 80 -j DNAT --to-destination \ 192.168.0.3-192.168.0.5

Caso especial do DNAT usado para redirecionar pacotes com destino de uma porta para uma porta local (por exemplo, para fazer proxys transparentes).

Similar ao SNAT, a configuração do DNAT é feita com o comando *iptables*, para inserir regras em uma chain da tabela NAT. As regras de DNAT devem ser aplicadas na chain PREROUTING, que é percorrida logo que os pacotes chegam à interface de rede, e antes da tomada de decisão de roteamento. O alvo utilizado nas regras é o DNAT, que deve ser utilizado com a opção *--to-destination* seguida do(s) IP(s) ou porta(s) a serem utilizados no DNAT. Alguns exemplos:

- Fazendo o DNAT dos pacotes que chegam com destino ao endereço;
- 200.0.0.1, da interface eth0, para o endereço 192.168.0.10:

```
# iptables -t nat -A PREROUTING -i eth0 -d 200.0.0.1 -j DNAT -to-
destination 192.168.0.10
```

- Balanceando requisições à porta 80, que chegam pela interface ppp0 para os servidores 192.168.0.3 a 192.168.0.5:

```
# iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 80 -j \ DNAT
-to-destination 192.168.0.3-192.168.0.5
```

- Fazendo o DNAT de conexões com destino ao endereço 200.0.0.1, porta 2222, para o endereço 192.168.0.10, porta 22:

```
# iptables - t nat -A PREROUTING -p tcp -d 200.0.0.1 --dport 2222 -j
DNAT --to-destination 192.168.0.10:22
```

Há um caso especial de DNAT, os chamados proxys transparentes. Neles, conexões de saída com destino à porta 80 são redirecionadas para um proxy local, sem que haja necessidade de o usuário configurar o seu navegador para isso. No netfilter, para esse tipo de DNAT é utilizado o alvo REDIRECT com a opção *--to-ports*. Note que é necessário indicar para esse alvo apenas a porta ou os intervalos de portas, já que o IP de destino sempre será o endereço local 127.0.0.1. O exemplo seguinte poderia ser utilizado para implementar um proxy transparente, redirecionando conexões remotas com destino à porta 80 para a porta local 3128:

```
# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j \
REDIRECT --to-ports 3128
```

Roteamento avançado

- Enlaces virtuais ponto-a-ponto.
- Encapsulamento de protocolo.
- No Linux, há os seguintes modos:
  - IPIP: IP sobre IP.
  - SIT: IPv6 sobre IP.
  - GRE: qualquer protocolo sobre IP.

Túneis são enlaces virtuais ponto-a-ponto, onde é feito o encapsulamento de pacotes, com o mesmo protocolo ou com protocolos diferentes, que são enviados sobre a infraestrutura IP. O uso de túneis permite o fluxo de pacotes entre hosts, independente do protocolo usado para conectá-los.

No Linux, desde a versão 2.2 do kernel, há os seguintes modos de túneis:

Modo	Descrição
ipip	IP sobre IP
sit	IPv6 sobre IP
gre	Qualquer protocolo GRE sobre IP

Tabela 8.1
Modos de túneis.

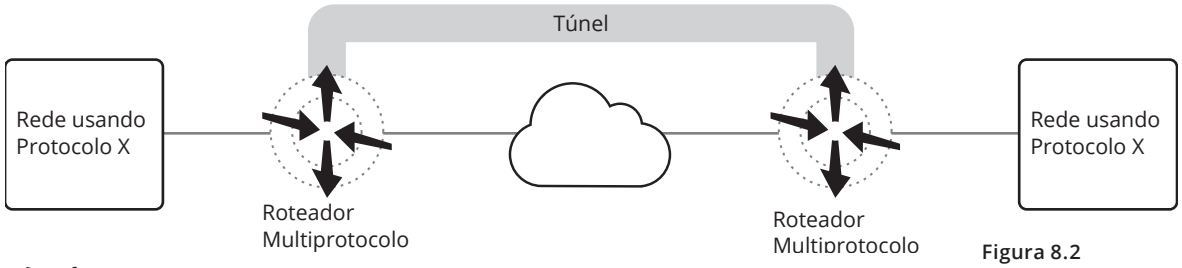


Figura 8.2
Rede usando o Protocolo X e roteadores Multiprotocolo.

Túneis

São divididos em duas classes:

- Point-to-point:
  - Possuem o endereço remoto e encaminham todos os pacotes para esse destino.
- Non-Broadcast Multi-Access (NBMA):
  - Não necessitam de um endereço remoto.
  - Uma rota é adicionada para a interface do túnel, informando o gateway.

Túneis são divididos em duas classes:

- **Point-to-point**: túneis que possuem um endereço remoto, destino para onde encaminham todos os pacotes;
- **Non-Broadcast Multi-Access (NBMA)**: túneis que não possuem um endereço remoto.

Depois que um túnel point-to-point é configurado, deve-se configurá-lo como se fosse um dispositivo qualquer de rede. Já para fazer roteamento com túneis NBMA, é preciso informar para aonde os pacotes devem ser encaminhados, ou seja, são criadas rotas com outro gateway informado para usar o túnel.

Comandos de configurações

Utilização do comando *ip*:

- Utiliza-se o objeto "tunnel".

Sintaxe:

ip tunnel add <NOME> mode <MODO> [local <S>] \ [remote <D>]

- Com o comando *ip*, o modo do túnel é informado como parâmetro.

Comando ip tunnel

Desde a versão 2.2 do kernel do Linux, os modos de túneis devem ser criados com o comando *ip*. O uso do comando *ip* permite informar o modo do túnel como um parâmetro.

A seguir são apresentadas as formas de se manusear túneis.

```
# ip tunnel help

Uso: ip tunnel { add | change | del | show } [ NAME ]

[ mode { ipip | gre | sit } ] [ remote ADDR ] [ local ADDR ]

[ [i|o]seq ] [ [i|o]key KEY ] [ [i|o]csum ]

[ ttl TTL ] [ tos TOS ] [ [no]pmtudisc ] [ dev PHYS_DEV ]
```

Onde:

```
NAME := STRING

ADDR := { IP_ADDRESS | any }

TOS := { NUMBER | inherit }

TTL := { 1..255 | inherit }

KEY := { DOTTED_QUAD | NUMBER }
```

O comando seguinte permite criar os diversos tipos de túneis, usando o comando *ip*:

```
# ip tunnel add <NOME> mode <MODO> [local <S>] [remote <D>]
```

- Este comando cria um novo túnel denominado de <NOME>, onde <NOME> é uma string arbitrária;
- O <MODO> seta o modo do túnel. Atualmente os três modos são ipip, sit e gre;
- Para informar o endereço local do túnel, após *local* é usado o endereço IP <S>;
- Para informar dados do endpoint do túnel, após *remote* é usado o endereço IP <D>.

Após ser criado, o túnel torna-se uma interface do host. E da mesma forma que ocorre com uma interface, é interessante testar a sua conectividade antes de se adicionar uma rota. Isso pode ser feito acrescentando endereços para a interface do túnel e após executar um ping. Exemplo:

```
# ip addr add <ip>/30 dev <NOME>
```

Túnel IP sobre IP (IPIP)

- Suporte no kernel (módulo IPIP).

Como fazer um túnel entre a rede A e B da figura abaixo? O pré-requisito para utilizar túneis no modo IPIP é ter suporte no kernel. Para isso, certifique-se de que o kernel do Linux tenha sido compilado com essa opção. Caso esteja compilado como módulo, deve-se adicionar o seguinte módulo:

```
# modprobe ipip
```

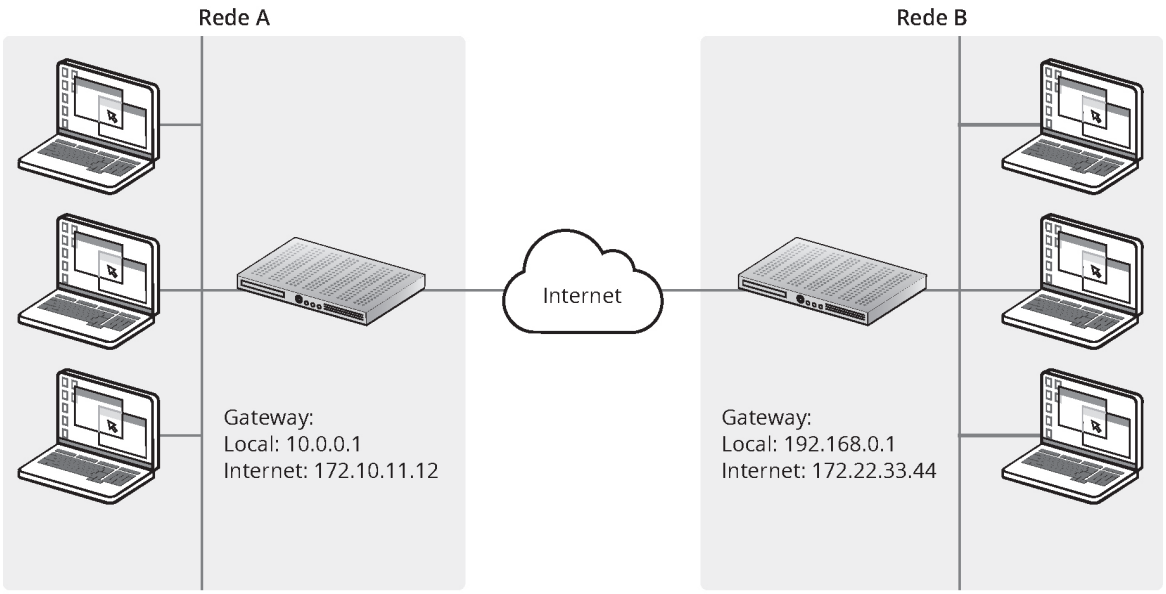


Figura 8.3
Tunelamento IPIP/GRE

Uso do túnel IPIP

Há duas redes, A e B. Ambas utilizam IP e no caminho delas há outra rede, por exemplo, a internet. É possível fazer um túnel entre a rede A e a rede B, para que os pacotes entre elas sejam encapsulados pelo túnel e as redes fiquem conectadas com possibilidade de uso da internet.

Como mostra a figura, suponha que a rede A tenha o prefixo 10.0.0.0/24, um gateway com endereço local 10.0.0.1 e um endereço na internet 172.10.11.12. Suponha, ainda, que a rede B tenha o prefixo 192.168.0.0/24, um gateway com endereço local 192.168.0.1 e um endereço na internet 172.22.33.44.

Será utilizado o prefixo 192.10.10.0/30 para endereçar o túnel. Para que toda a rede A tenha acesso à rede B através do túnel, o gateway da rede A deve ser configurado da seguinte forma:

```
#ip tunnel add redeb mode ipip remote 172.22.33.44 local
172.10.11.12 ttl 255

#ip link set redeb up

#ip addr add 192.10.10.1/30 dev redeb

#ip route add 192.168.0.0/24 dev redeb

O gateway da rede B deve ter configuração análoga:

#ip tunnel add redea mode ipip remote 172.10.11.12 local
172.22.33.44 ttl 255

#ip link set redea up

#ip addr add 192.10.10.2/30 dev redea

#ip route add 10.0.0.0/24 dev redea
```

O exemplo a seguir criará o túnel da figura anterior, utilizando o comando *ifconfig*:

Gateway da rede A:

```
# ip tunnel add redeb mode ipip remote \ 172.22.33.44 local
172.10.11.12 ttl 255
# ip link set redeb up
# ip addr add 192.10.10.1/30 dev redeb
# ip route add 192.168.0.0/24 dev redeb
```

Como ficaria a configuração do gateway da rede B?

Túnel GRE

- O túnel Generic Routing Encapsulation foi originalmente desenvolvido pela Cisco.
- Tornou-se um padrão RFC1701/1702.
- Usado largamente por permitir tunelamento de uma grande variedade de protocolos.
- Também necessita de suporte no kernel (módulo ip_gre).

Como fazer um túnel entre as redes A e B da figura anterior? Da mesma forma que o modo de túnel IPIP, o pré-requisito para utilização do modo GRE no Linux é ter suporte no kernel. Para isso, certifique-se de que o kernel tenha sido compilado com essa opção. Caso esteja compilado como módulo, deve-se adicionar o seguinte módulo:

```
# modprobe ip_gre
```

Use o mesmo exemplo do item anterior. Para criar um túnel modo GRE entre as redes A e B, usando o comando *ip*, a configuração do gateway da rede A será feita da seguinte forma:

```
#ip tunnel add redeb mode gre remote 172.22.33.44 local 172.10.11.12
ttl 255
#ip link set redeb up
#ip addr add 192.10.10.1/30 dev redeb
#ip route add 192.168.0.0/24 dev redeb
```

O gateway da rede B deve ter configuração análoga:

```
#ip tunnel add redea mode gre remote 172.10.11.12 local 172.22.3
3.44 ttl 255
#ip link set redea up
#ip addr add 192.10.10.2/30 dev redea
#ip route add 10.0.0.0/24 dev redea
```

O exemplo a seguir vai criar o túnel da figura anterior, utilizando o comando *ip:*

- Gateway da rede A:

```
#ip tunnel add redeb mode gre remote\ 172.22.33.44 local
172.10.11.12 ttl 255
#ip link set redeb up
#ip addr add 192.10.10.1/30 dev redeb
#ip route add 192.168.0.0/24 dev redeb
```

- Gateway da rede B:

```
#ip tunnel add redea mode gre remote\ 172.10.11.12 local
172.22.33.44 ttl 255
#ip link set redea up
#ip addr add 192.10.10.2/30 dev redea
#ip route add 10.10.0.0/24 dev redea
```

Túnel SIT

- Utilizado para conectar domínios IPv6 através do protocolo IPv4.
- Necessidade de suporte IPv6 no kernel para fazer túneis SIT.
- Uso semelhante aos modos IPIP e GRE, mas prefixo da rede de destino é IPv6 ao invés de IPv4.

Os túneis no modo SIT conectam ilhas que operam com o novo protocolo IPv6 através do protocolo IPv4. Isso ocorre porque túneis precisam ser criados em domínios que utilizem o novo protocolo para se comunicar com os demais domínios IPv6 no mundo, porém utilizando o protocolo IP atual (IPv4).

Diferente dos outros modos de túneis, o modo SIT está disponível no Linux se o kernel possuir suporte para IPv6. Da mesma forma que os outros modos, o suporte ao IPv6 pode ser um módulo ou estar construído junto ao kernel. Nas versões mais recentes do Linux (>2.4) o suporte ao IPv6 está disponível no módulo denominado IPv6:

```
# modprobe ipv6
```

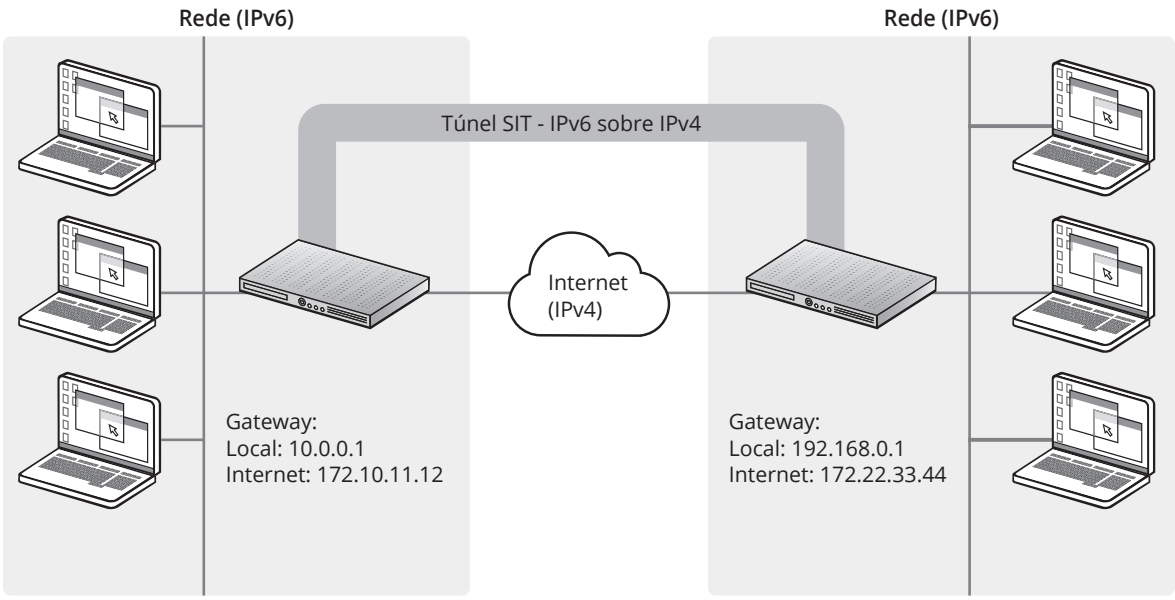


Figura 8.4
Uso do túnel SIT.

A configuração dos túneis SIT é análoga aos modos IPIP e GRE já vistos, mas as rotas que vão utilizar o túnel devem possuir o prefixo da rede em IPv6 em vez de em IPv4:

```
# ip tunnel add <nome> mode sit remote <endereçoV4remoto> local
<endereçoV4local>

# /sbin/ip link set dev <nome> up

# ip -6 route add <prefixoV6> dev <nome>
```

Tabelas de roteamento

- O Linux tem suporte a múltiplas tabelas de roteamento.
- Permite criar políticas de roteamento.
- Informações sobre as tabelas (/etc/iproute2/rt_tables).

O suporte para as múltiplas tabelas de roteamento existe desde a versão 2.2 do kernel do Linux. O sistema de múltiplas tabelas de roteamento permite ao administrador uma infraestrutura flexível para implementar políticas de roteamento. O uso de tabelas de roteamento extras é utilizado junto com Routing Policy Database (RPDB), que será visto adiante. Isso permite, por exemplo, fazer roteamento pela origem do endereço IP ou por diversos outros critérios de escolha.

As informações sobre as tabelas de roteamento estão descritas no arquivo */etc/iproute2/rt_tables*.

As tabelas são identificadas por um valor entre 1 e 255 e por um nome. O conteúdo do arquivo */etc/iproute2/rt_tables* é:

```
#
# reserved values
#
255 local
```

```
254main
253default
0  unspec
#
# local
#
1  inr.ruhep
```

Na tabela seguinte são explicadas as linhas desse arquivo.

Identificador	Nome	Descrição
255	local	Tabela de roteamento especial mantida pelo kernel. Usuários não podem adicionar novas entradas nessa tabela.
254	main	A principal tabela de roteamento. Nela ficam as rotas quando adicionadas com o comando route ou ip route (sem especificar a tabela).
253	default	Tabela especial utilizada pelo kernel.
0	unspec	Operações sobre essa tabela se refletem em todas as demais tabelas.
1	Inr.ruhep	Essa tabela é um exemplo indicando que a tabela 1 tem o nome inr.ruhep.

Caso esteja implementando alguma política de roteamento que necessite de alguma tabela extra, para que ela seja adicionada basta editar o arquivo *rt_tables* e adicionar uma nova linha, um identificador e um nome. É possível adicionar até 252 tabelas de roteamento.

Tabela 8.2
Linhas do arquivo */etc/iproute2/rt_tables*.

Entradas na tabela de roteamento

Cada tabela pode ter várias entradas, manipuladas com o comando *ip route*:

- unicast.
- broadcast.
- local.
- nat.
- unreachable.
- prohibit.
- blakchole.
- throw.

Cada tabela de roteamento pode ter várias rotas que podem ser manipuladas pelo administrador com o comando *ip route*. Cada rota na tabela de roteamento possui características próprias que permitem diferentes comportamentos e funcionalidades.

- **Rota unicast**: a mais comum na tabela de roteamento. Ela é assumida sempre que não é informado o tipo de rota ao comando *ip*. Exemplos:

```
# ip route add unicast 192.168.0.0/24 via 192.168.100.5
# ip route add default via 193.7.255.1
```

- **Rota broadcast**: usada pelos dispositivos da camada de enlace. É usada na tabela local e tipicamente manuseada pelo próprio kernel. Exemplo:

```
# ip route add table local broadcast 10.10.20.255 dev eth0 proto
kernel scope link src 10.10.20.67
```

- **Rota local**: o kernel adicionará uma entrada para a tabela de roteamento local quando o endereço IP for adicionado para uma interface. Exemplo:

```
# ip route add table local local 10.10.20.64 dev eth0 proto kernel
scope host src 10.10.20.67
```

- **Rota nat**: adicionada ao kernel quando é configurado stateless NAT (trocar o destino do pacote). Exemplo:

```
# ip route add nat 192.168.0.1 via 172.16.82.184
```

- **Rota unreachable**: diz que o destino da rota é inalcançável e vai gerar um ICMP unreachable para o remetente. Exemplo:

```
# ip route add unreachable 172.16.82.184
```

- **Rota prohibit**: funciona de modo semelhante à rota unreachable, mas gera um ICMP prohibit para o remetente. Exemplo:

```
# ip route add prohibit 10.21.82.157
```

- **Rota blackhole**: descarta os pacotes que a utilizarem. Exemplo:

```
# ip route add blackhole 202.143.170.0/24
```

- **Rota throw**: útil quando utilizada com uma política de roteamento, pois faz com que o pacote que a use falhe e continue sendo analisado pelo RPDB. Exemplo:

```
# ip route add throw 10.79.0.0/16
```

Como já visto, é possível manipular as rotas da tabela de roteamento com o comando *ip route*. Quando não é informada uma tabela de roteamento, ao se adicionar uma rota é utilizada a tabela main. Porém, pode ser necessário informar ao comando a tabela de roteamento. Por exemplo, suponha que no arquivo */etc/iproute2/rt_tables/* foi adicionada uma nova tabela, chamada "cliente1" e identificada por 252. Para adicionar uma rota default nesta tabela, é usado o seguinte comando:

```
# ip route add table cliente1 default via <IP>
```

Ao se consultar a tabela de roteamento com o comando *ip*, a tabela exibida é a main. Portanto, para se consultar uma outra tabela de roteamento (a tabela "cliente1" do exemplo anterior), é necessário informar a tabela:

```
# ip route show table cliente1
```

No Linux, um cache de roteamento implementado no kernel armazena as consultas recentes feitas nas tabelas de roteamento. Este cache permite que as demais consultas feitas sejam referenciadas mais rapidamente. Para consultar este cache, pode ser utilizado o comando:

```
# ip route show cache
```

Este cache expira periodicamente. Para adiantar esse processo e garantir o comportamento esperado, ao se fazer uma alteração nas tabelas de roteamento é importante limpar este cache, com o seguinte comando:

```
# ip route flush cache
```

Routing Policy Database (RPDB)

- Usada pelo kernel para controlar a ordem de consulta nas diversas tabelas.
- O RPDB é manipulado pelo comando *ip rule*.
- Cada regra possui uma prioridade, que é examinada sequencialmente (0 a 32767).

O banco de dados da política de roteamento (Routing Policy Database – RPDB) é utilizado pelo kernel para controlar a ordem de consultas nas diversas tabelas de roteamento. Antes de um pacote ser roteado por uma entrada de alguma tabela, o pacote consulta o RPDB para decidir a política de roteamento.

O administrador pode querer manipular a RPDB para adicionar regras que façam com que um determinado pacote IP utilize uma tabela de roteamento diferente. Estas regras são manuseadas com o comando *ip rule*.

Cada regra adicionada possui uma prioridade: são examinadas sequencialmente, da regra identificada com 0 até a regra 32767.

```
# ip rule show

0:  from all lookup local

32766: from all lookup main

32767: from all lookup default
```

Note que a tabela de roteamento main possui prioridade 32766. As regras que o administrador vai inserir deverão ter uma prioridade maior (identificador menor) para que tenham efeito. Portanto, quando uma regra é adicionada, por padrão ela possui prioridade maior (identificador menor) do que uma regra que já existe no RPDB. Podemos, contudo, informar a preferência de uso da regra da seguinte forma:

```
# ip rule add unicast from 10.0.0.0/24 table cliente1 pref <valor>
```

Semelhantes às do comando *ip route*, as regras que podem ser adicionadas pelo administrador possuem características próprias. Essas características permitem diferentes comportamentos e funcionalidades.

- **Regra unicast**: tipo mais comum, informa o kernel para usar uma determinada tabela de acordo com a regra. Exemplo:

```
# ip rule add unicast from 192.168.100.0/24 table 5
```

- **Regra nat**: corrige operações de stateless NAT (rescrever a origem do pacote). Exemplo:

```
# ip rule add nat 192.168.0.1 from 172.16.82.184
```

- **Regra unreachable**: torna inatingível qualquer pacote que case com ela, gerando um ICMP unreachable. Exemplo:

```
# ip rule add unreachable from 192.168.7.0/25
```

- **Regra prohibit**: funciona de forma semelhante à regra unreachable, mas gera um ICMP prohibit para o remetente. Exemplo:

```
# ip rule add prohibit from 192.168.12.0
```

- **Regra blackhole**: descarta qualquer pacote que case com ela. Exemplo:

```
# ip rule add blackhole from 209.10.26.51
```

Após ter sido estudado como adicionar tabelas, rotas e regras de roteamento, é possível, por exemplo, utilizar uma tabela de roteamento selecionada por uma regra do RPDB para fazer um roteamento pela origem. A regra do RPDB informará ao kernel que os pacotes com uma determinada origem utilizam uma tabela de roteamento adicionada pelo administrador:

```
# ip rule add from 10.0.0.0/24 table <tabela>
```

A tabela de roteamento adicionada deve possuir alguma rota que informe para onde enviar o pacote:

```
# ip route add default via <gateway> table <tabela>
```

Para que a configuração comece a funcionar sem precisar aguardar o cache expirar, o administrador pode expirar a cache do kernel manualmente:

```
# ip route flush cache
```

Tipos de regras

- unicast.
- nat.
- unreachable.
- prohibit.
- blackhole.

A última regra adicionada possui mais prioridade. Pode-se informar o valor da prioridade explicitamente.

```
# ip rule add unicast from 10.0.0.0/24 table\ cliente1 pref <valor>
```

Com RPDB e múltiplas tabelas, como podemos fazer roteamento pela origem?

```
# ip rule add from 10.0.0.0/24 table \ <tabela>
# ip route add default via <gateway> table \ <tabela>
# ip route flush cache
```

Roteiro de Atividades 8

Atividade 8.1 – SNAT

Crie uma regra de SNAT para que os pacotes do tipo ICMP saiam da sua máquina com o endereço de origem trocado para 10.0.0.1. Utilize os comandos *ping* e *tcpdump* para verificar o que acontece ao se tentar pingar um host. Remova a regra após terminar a atividade.

Atividade 8.2 – SNAT (continuação)

Utilize dois terminais ou consoles para esta atividade. Em um deles, execute um *tcpdump*; em outro, faça o seguinte:

1. Um telnet para a porta 80 de um site qualquer. Qual foi a porta de origem utilizada pelo sistema?
2. Crie uma regra de SNAT para que os pacotes com destino à porta 80 saiam com a porta de origem entre os valores 6000 e 7000.
3. Repita o item (a). Qual foi a porta de origem dos pacotes que saíram da máquina local?
4. Remova as regras de NAT.

Atividade 8.3 – DNAT

As Atividades 8.3, 8.4, 8.5, 8.6 e 8.7 devem ser feitas em duplas, com cada aluno ou grupo em uma máquina.

Utilize dois terminais ou consoles para esta atividade. Em um deles, execute um *tcpdump*; em outro, faça o seguinte:

1. Crie uma regra de DNAT para que todos os pacotes com origem na máquina de sua dupla, e com endereço de destino, IP e porta de destino 8080, sejam redirecionados para o endereço de um servidor http na porta 80 (por exemplo, <ip_google>:80). Não se esqueça de habilitar o IP forward.
2. A partir de outra máquina, execute um telnet para o endereço de sua máquina, na porta 8080. O que acontece? É possível completar a conexão? Explique.

Atividade 8.4 – DNAT (continuação)

O que é necessário fazer para permitir que as conexões do item (b) da Atividade 8.3 sejam completadas? Teste a sua solução.

Atividade 8.5 – Redirect

Nesta atividade será explorado o alvo REDIRECT do netfilter. Combine com o seu colega quem será o servidor e quem será o cliente. Execute:

1. No servidor, execute o comando *nc* no modo listen na porta 3000. A partir da máquina cliente, teste com o comando *nc* se é possível conectar na porta 3000 do servidor.
2. No servidor, crie uma regra para redirecionar para a porta 3000 as conexões com destino à porta 6000. A partir do cliente, teste uma conexão com destino à porta 6000 do servidor.

Atividade 8.6 – Túnel IPIP

Combine com a sua dupla quem será a ponta A e quem será a ponta B:

1. Carregue o módulo IPIP. Em seguida, da máquina A para B (e vice-versa), crie um túnel chamado "meutunel" com o comando *ip tunnel*.
2. Adicione o endereço 10.0.0.1/30 à máquina A, e o endereço 10.0.0.2/30 à máquina B. Utilize o comando *ip addr*.
3. Levante as interfaces do túnel com o comando *ip link*.
4. A partir da máquina A, faça um ping para 10.0.0.2. Utilize o *tcpdump* e verifique como esses pacotes são vistos nas interfaces "meutunel" e eth0.

Atividade 8.7 – MTU

Sobre o túnel feito na atividade anterior, responda:

1. Qual é a MTU utilizada na interface do túnel?
2. O que pode acontecer com pacotes de tamanho maior que a MTU do túnel?
3. Onde esses pacotes serão remontados?

Atividade 8.8 – Tabela de Roteamento

Sobre tabelas de roteamento, verifique:

1. Quais tabelas de roteamento existem na sua máquina?
2. As rotas da tabela main (dica: use o comando *ip route*).
3. As rotas da tabela local.

Atividade 8.9 – Testando rotas

As atividades 8.9 e 8.10 devem ser feitas em duplas, com cada aluno ou grupo em uma máquina.

Uma máquina fará o papel de roteador e outra será utilizada para testar as rotas. Utilize o comando *ip route* para:

1. Criar no roteador uma rota do tipo *unreachable* para o destino 10.0.100.0/24. Na outra máquina, crie uma rota para essa rede utilizando como gateway o endereço do roteador. Execute um *ping* para o endereço 10.0.100.1 e verifique o comportamento com o *tcpdump*.
2. Criar uma rota do tipo *prohibit* para o destino 10.0.101.0/24. Na outra máquina, crie uma rota para essa rede utilizando como gateway o endereço do roteador. Execute um *ping* para o endereço 10.0.101.1 e verifique o comportamento com o *tcpdump*.
3. Criar uma rota do tipo *blackhole* para o destino 10.0.102.0/24. Na outra máquina, crie uma rota para essa rede utilizando como gateway o endereço do roteador. Execute um *ping* para o endereço 10.0.102.1 e verifique o comportamento com o *tcpdump*.

Atividade 8.10 – Roteamento pela origem

Esta atividade utiliza uma técnica chamada roteamento pela origem.

Uma máquina fará o papel de roteador e outra será utilizada para testar as rotas.

1. No roteador, crie duas tabelas de roteamento: a tabela “buraconegro”, com prioridade 5, e a tabela “proibida”, com prioridade 6. Dica: edite o arquivo */etc/iproute2/rt_tables*.
2. No roteador, acrescente uma rota default do tipo *blackhole* na tabela “buraconegro”.
3. No roteador, acrescente uma rota default do tipo *prohibit* na tabela proibida.
4. Na outra máquina, acrescente uma rota para 10.0.200.0/24 via <ip_roteador>.
5. No roteador, acrescente uma regra para que tudo que venha do outro host seja roteado utilizando-se a tabela “buraconegro”. No host, execute um *ping* 10.0.200.1. O que acontece?
6. No roteador, remova a regra do item (e) e acrescente uma regra para que tudo que venha do host seja roteado com a tabela proibida. Execute o comando *ip route flush cache* em ambas as máquinas. No host, execute um *ping* 10.0.200.1. O que acontece?

9

VPN e protocolos de tunelamento de nível 2

objetivos

Entender o conceito de Virtual Private Network (VPN) e sua aplicação; entender o que é tunelamento, e configuração e instalação de OpenVPN.

conceitos

VPN, pacotes LCP, protocolos SLIP, GRE, PPTP, L2F, L2TP, IPSec e tunelamento.

Virtual Private Network (VPN)

Dividida em duas categorias:

- Acesso remoto.
- Intranet ou extranet.

Vantagens:

- Custo mais baixo que em redes privadas.
- Flexibilidade maior.
- Menor esforço de gerenciamento.
- Simplicidade de topologia de rede.

VPN é uma forma de comunicação utilizada por uma rede corporativa apoiada numa infraestrutura pública compartilhada, empregando a mesma segurança, gerenciamento e políticas de desempenho aplicadas numa rede privada. Pode ser dividida em duas categorias, cada uma com diferentes características de segurança e desempenho:

- **Acesso remoto (remote access)**: conectam usuários móveis com uma quantidade pequena de tráfego para a rede corporativa;
- **Intranet ou extranet**: conecta locais fixos, filiais e escritórios pertencentes a uma WAN corporativa.

Vantagens de uma VPN para uma rede corporativa:

- **Custo mais baixo do que o de redes privadas**: reduz o custo de operação, o equipamento de backbone e a largura de banda da rede de transporte.
- **Flexibilidade maior**: permite aproveitar a economia da internet, oferecendo maior flexibilidade às mudanças necessárias para acompanhar a demanda e a evolução dos negócios.

- **Menor esforço de gerenciamento**: VPN é mais simples de operar e gerenciar do que uma rede privada proprietária.
- **Simplicidade de topologia de rede**: a utilização de um backbone IP elimina a necessidade de uso de protocolos orientados à conexão, tais como Frame Relay e ATM.

Proteção da VPN

- Túneis e encriptação.
- Autenticação de pacotes.
- Firewall e detecção de intrusos.
- Autenticação de usuários.

A utilização de infraestrutura de rede WAN pública compartilhada traz alguns problemas de segurança bastante complexos. As empresas precisam garantir que suas VPNs são seguras contra ataques, tentativas de invasão e acessos não autorizados. Para proteção adequada da VPN devemos ter:

- **Túneis e encriptação**: túneis criptografados protegem os dados de interceptação e visualização por usuários não autorizados, executando encapsulamento multiprotocolo, quando necessário;
- **Autenticação de pacotes**: a integridade dos dados numa rede compartilhada é também um requisito de segurança. Numa rede insegura, pacotes podem ser interceptados e seu conteúdo alterado. A autenticação de pacotes protege contra esse tipo de violação de segurança utilizando cabeçalhos adicionais ao pacote IP (o AH do IPSec, por exemplo);
- **Firewall e detecção de intrusos**: o firewall monitora o tráfego que cruza o perímetro da rede e impõe restrições de acordo com uma política de segurança, além de proteger contra ataques de rede. O sistema de detecção de intruso opera em conjunto com o firewall analisando o conteúdo e o contexto dos pacotes para determinar se o tráfego é autorizado e para verificar se a política de segurança está sendo cumprida no âmbito da rede corporativa;
- **Autenticação de usuário**: permite que apenas usuários autorizados tenham acesso aos recursos da rede, enquanto os não autorizados devem ser banidos;

Uma VPN deve assegurar o uso eficiente da largura de banda e desempenho confiável para dados importantes. A natureza do tráfego de dados, devido à sua característica de rajadas, tende a deixar ociosos os enlaces por alguns períodos de tempo e em outros momentos cria congestionamentos pelo envio excessivo de pacotes de uma só vez.

Arquitetura VPN de acesso remoto

VPN de acesso remoto estende as facilidades de rede corporativa a usuários remotos em geral. Os usuários poderão se conectar às intranets ou **extranets** quando e como quiserem.

Extranet
É a porção da rede de computadores de uma empresa que faz uso da internet para partilhar com segurança parte do seu sistema de informação.

Numa arquitetura VPN de acesso remoto, tunelamento e encriptação podem ser iniciados no cliente PC ou no Network Access Server (NAS). No primeiro caso, o túnel criptografado é estabelecido no cliente, usando IPSec, L2TP ou PPTP.

No caso do NAS, não existem requisitos especiais no software do cliente. O usuário remoto disca para o Ponto de Presença (POP) do provedor de serviço usando uma conexão PPP/SLIP, é autenticado pelo provedor do serviço e inicia um túnel seguro do POP até a rede corporativa onde o usuário é novamente autenticado. A desvantagem é que o trecho entre o cliente e o POP não é seguro.

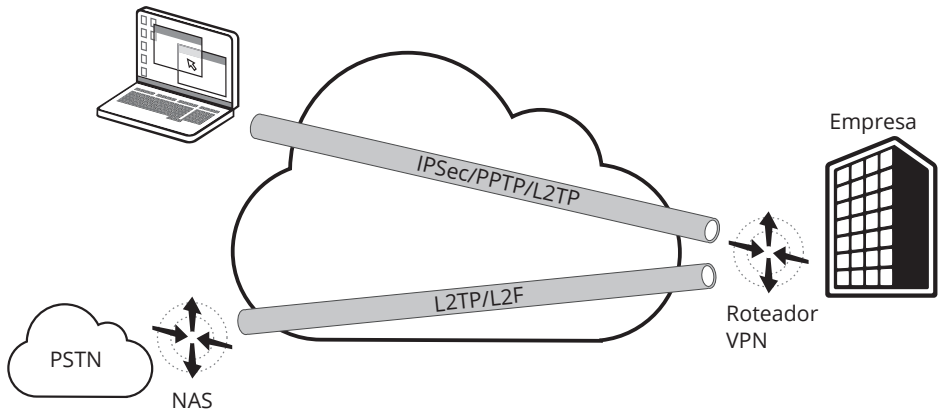


Figura 9.1
A Arquitetura VPN de acesso remoto.

Arquitetura VPN intranet e extranet

VPNs intranet/extranet substituem as linhas privadas ou outra infraestrutura de WAN por infraestrutura de rede compartilhada (como a internet) ou fornecida por provedores de serviço, tais como redes IP, Frame Relay ou ATM.

VPNs intranet usam IPSec ou GRE para a criação de túneis seguros através da rede. Quando a internet é usada, é preciso lembrar que não há garantia de QoS, embora os custos sejam relativamente mais baixos.

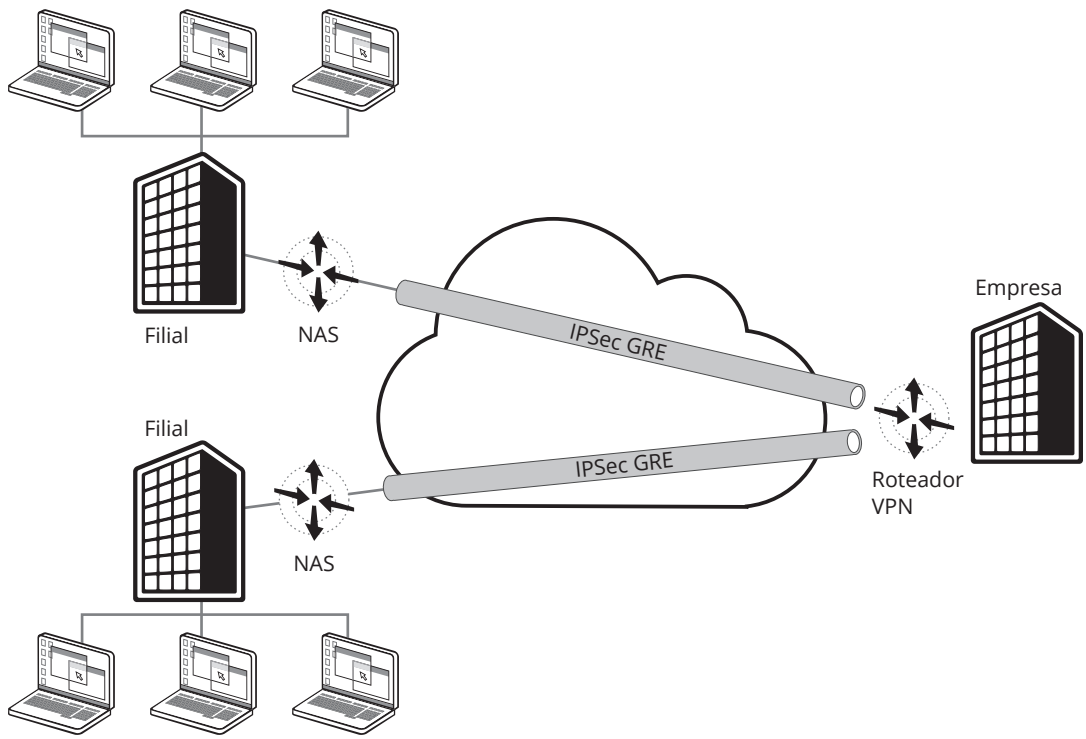


Figura 9.2
Arquitetura VPN intranet e extranet.

Protocolos de tunelamento – PPP/SLIP

- Linhas seriais ponto-a-ponto.
- Encapsulam pacotes IP em linhas seriais.
- Componentes do PPP:
  - High-Level Data Link Control (HDLC).
  - Link Control Protocol (LCP).
  - Network Control Protocol (NCP).
- Interfaces DTE/DCE, circuitos duplex.

Protocolo PPP

A grande maioria dos hosts na internet era conectada através de redes locais de vários tipos, sendo a Ethernet a mais comum. Outros hosts eram conectados através de redes X.25 e relativamente poucos através de linhas seriais ponto-a-ponto. Uma razão para isso era a falta de um protocolo padrão de encapsulamento internet. O PPP foi concebido para resolver esse problema.

Assim, o PPP provê um padrão de encapsulamento do protocolo IP sobre linhas seriais. Além disso, ele permite atribuição e gerenciamento de endereços IP, encapsulamento assíncrono (startlstop) e síncrono orientado a bit (bit-oriented), multiplexação de protocolos de rede, configuração de linhas, teste da qualidade da linha, detecção de erros e, opcionalmente, negociação de endereços da camada de rede e compressão de dados.

Componentes do PPP:

- HDLC para encapsulamento de datagramas sobre linhas seriais;
- LCP para o estabelecimento, configuração e teste da conexão de Enlace de Dados;
- NCP para negociação das opções de múltiplos protocolos de camada de rede.

01111110	11111111	00000011	Protocolo	Informação	Verificação	01111110
Flag	Endereço	Controle	1 ou 2 bytes	Comp. variável	1 ou 2 bytes	Flag

Figura 9.3
Formato do quadro PPP.

O protocolo PPP é capaz de operar com qualquer interface DTE/DCE. Exemplos: RS-232-C, RS-422, RS-423 e V.35. A única exigência absoluta é a provisão de um circuito duplex, dedicado ou comutado, que possa operar no modo bit-serial síncrono ou assíncrono, transparente para os quadros de Enlace de Dados PPP. Não há restrição quanto à velocidade dos enlaces, exceto a limitação das interfaces.

O quadro PPP mostrado na figura é composto pelos seguintes campos:

- **Flag**: 1 byte que indica o início e o fim do quadro; sequência binária 01111110;
- **Address**: 1 byte com a sequência binária 11111111 (endereço broadcast padrão); o protocolo PPP não atribui endereço individual à estação;
- **Control**: 1 byte com a sequência binária 00000011, que sinaliza transmissão de dados de usuário em quadro não sequenciado; é um serviço de Enlace de Dados sem conexão similar ao LLC tipo 1;

- **Protocol**: 2 bytes que identificam o protocolo encapsulado no campo de informação do quadro;
- **Data**: zero ou mais bytes que contêm o datagrama do protocolo (PDU) especificado no campo de protocolo; o final do campo de dados é indicado pelo campo *Flag* e permite ainda 2 bytes para o campo FCS; o tamanho máximo default do campo de dados é 1.500 bytes, embora implementações PPP possam usar outros valores, através de negociação;
- **Frame Check Sequence (FCS)**: 2 bytes por default; podem ser usados 4 bytes para melhor detecção de erros.

O PPP Link Control Protocol (LCP) fornece um método para o estabelecimento, manutenção e terminação das conexões ponto-a-ponto. O LCP tem 4 fases:

- **Estabelecimento do enlace e negociação da configuração**: antes de iniciar a transmissão de qualquer PDU do protocolo da camada de rede (exemplo: IP), LCP precisa abrir a conexão e negociar os parâmetros de configuração; essa fase se encerra quando um quadro de reconhecimento de configuração é enviado e recebido.
- **Determinação da qualidade do enlace**: LCP permite uma fase opcional de determinação da qualidade do enlace após a fase anterior; nessa fase o enlace é testado para determinar se a qualidade é suficiente para atender ao protocolo da camada de rede;
- **Negociação da configuração do protocolo da camada de rede**: após o LCP encerrar a fase anterior, os protocolos da camada de rede podem ser configurados separadamente e ativados ou desativados a qualquer tempo; se o LCP encerrar o enlace, ele informa ao protocolo da camada de rede para que possa tomar as ações apropriadas;
- **Terminação do enlace**: o LCP pode terminar o enlace a qualquer tempo; o encerramento normalmente é feito a pedido do usuário, mas pode ser feito devido à ocorrência de anormalidades.

O LCP possui 3 classes de quadros:

- Quadros de estabelecimento do enlace;
- Quadros de terminação do enlace;
- Quadros de manutenção do enlace.

Consulte o RFC 1661 na internet e saiba mais também em "Internetworking Technology Handbook", no site da Cisco.

O Network Control Protocol (NCP) é específico para cada protocolo de camada de rede e permite que as solicitações de configuração a serem feitas sejam específicas para o protocolo em questão. No caso do IP, por exemplo, a atribuição dinâmica de endereço IP é a possibilidade mais importante.

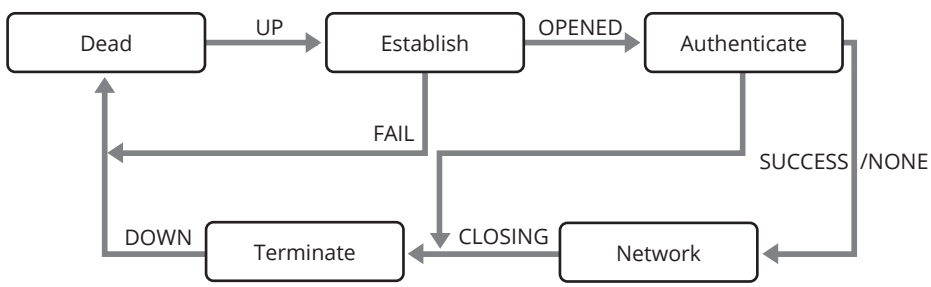


Figura 9.4
Protocolo PPP: usuário remoto com um computador.

Pacotes LCP

Onze tipos de pacotes LCP são definidos na RFC 1661 e estão na tabela a seguir:

Nome	Direção	Descrição
Configure-request	IR	Lista de opções e valores propostos.
Configure-ack	IR	Todas as opções são aceitas.
Configure-nak	IR	Algumas opções não são aceitas.
Configure-reject	IR	Algumas opções não são negociáveis.
Terminate-request	IR	Solicita a desativação da linha.
Terminate-ack	IR	Solicita a desativação da linha.
Code-reject	IR	Solicitação desconhecida recebida.
Protocol-reject	IR	Protocolo desconhecido solicitado.
Echo-request	IR	Favor enviar este quadro de volta.
Echo-reply	IR	Aqui está o quadro de volta.
Discard-request	IR	Só descarta este quadro (para fins de teste).

Tabela 9.1
Os pacotes LCP: onze no total.

Os quatro tipos de pacotes Configure permitem que o iniciador (I) proponha valores de opção e que o respondedor (R) os aceite ou rejeite. Nesse último caso, o respondedor poderá fazer uma proposta alternativa ou informar que não deseja negociar determinadas opções. As opções que estiverem sendo negociadas e seus valores propostos fazem parte do LCP.

Os códigos de Terminate são utilizados para encerrar uma linha quando ela não é mais necessária. Os códigos de Code-reject e Protocol-reject são utilizados pelo respondedor para indicar que ele recebeu algo que não consegue entender. Essa situação pode significar que ocorreu um erro de transmissão não detectado, mas provavelmente significa que o iniciador e o respondedor estão executando diferentes versões do protocolo LCP.

Os tipos de Echo são utilizados para testar a qualidade da linha. Por fim, utiliza-se o Discard-request para depuração de erros. Se uma das extremidades estiver tendo problemas em obter bits do cabo, o programador poderá utilizar esse tipo para teste.

Se conseguir terminar, ele será simplesmente descartado pelo receptor e não será executada qualquer outra ação.

As opções que podem ser negociadas incluem a definição do tamanho máximo da carga útil (payload) para os quadros de dados, a ativação da autenticação e a escolha do protocolo a ser utilizado, a ativação da monitoração da qualidade da linha durante a operação normal e a seleção de diversas opções de compactação de cabeçalhos.

De modo geral, há muito pouco a ser dito sobre os protocolos NCP. Cada um deles é específico para algum protocolo de camada de rede e permite que as solicitações de configuração a serem feitas sejam específicas para o protocolo em questão. No caso do IP, por exemplo, a atribuição dinâmica de endereços IP é a possibilidade mais importante.

Protocolo SLIP

Serial Line Internet Protocol (SLIP):

- Conexões seriais ponto-a-ponto.
- Coloca o datagrama IP em quadros com trailer END.
- Enlaces seriais dedicados ou discados de baixa velocidade.
- Pacotes no máximo de 1006 bytes.
- Simples e sem recursos.
  - Integra a RFC 1055.

A família de protocolos TCP/IP opera sobre uma grande variedade de sub-redes: IEEE 802.3 (Ethernet), 802.5 (Token Ring), linhas X.25, enlaces de satélite e linhas seriais. Existiam padrões para o encapsulamento de pacotes IP para todas essas sub-redes, exceto para linhas seriais. O primeiro protocolo definido para o encapsulamento de datagramas IP foi o SLIP – Serial Une IP, comumente usado para conexões seriais ponto-a-ponto rodando TCP/IP.

SLIP é simplesmente um protocolo de inserção de pacotes IP em quadros. SLIP define uma sequência de caracteres no quadro que carrega o pacote IP e nada mais. Não provê endereçamento, identificação do tipo de pacote ou mecanismos de detecção ou correção de erros ou de compressão. Por fazer tão pouco, é tão fácil de implementar. Em princípio, o SLIP envia um pacote com um caracter END no final.

As implementações do SLIP existentes limitam-se a pacotes de até 1006 bytes, sendo esse o tamanho máximo de datagrama recomendado. O RFC 1055 de 01/06/1988 define as características do SLIP e ainda mostra um código em C como exemplo de implementação.

Operação:

- Caracteres *end* (192) e *slip esc* (219).
- Envia o datagrama, termina com *end* .
- Se aparecer um *end*, envia ESC+220 .
- Se aparecer um ESC, envia ESC+221.

Problemas:

- Endereços conhecidos nas pontas.
- Somente um protocolo de cada vez.
- Sem detecção ou correção de erros.
- Sem compressão.

O protocolo SLIP define dois caracteres especiais: END e ESC. END é decimal 192 e ESC é decimal 219. Note que o caractere ESC nada tem a ver com ASCII ESC; para nossos propósitos chamaremos de SLIP ESC.

Para enviar um pacote, SLIP simplesmente inicia enviando os dados do pacote. Se um byte de dados for igual a um caracter END, ele envia uma sequência de dois bytes: ESC e decimal 220. Se um byte de dados for igual a ESC, ele envia uma sequência de dois bytes: ESC e decimal 221. Após o último byte do pacote, um caracter END é enviado.

Na época de sua definição o SLIP foi bastante útil. Hoje em dia foi largamente superado pelo protocolo PPP, principalmente devido às seguintes deficiências:

- Ambos os hosts nas duas pontes precisam conhecer o endereço um do outro; numa linha dedicada não há problema, mas numa linha discada será preciso implementar um sistema para a troca de informações de endereços, porque o SLIP não possui mecanismo para isso.
- Não existe campo para tipo de protocolo, assim apenas um protocolo de cada vez pode ser encaminhado na conexão serial, não sendo possível multiplexar protocolos no enlace.
- Não existem mecanismos de detecção ou correção de erros.
- Não existem mecanismos de compressão.

Protocolo GRE

Generic Routing Encapsulation (GRE):

- Procotolo utilizado, normalmente, em roteadores de borda;
- Desvantagens:
  - Configurado manualmente.
  - Quanto maior for a quantidade de túneis, maior será o consumo de banda.

Protocolo de tunelamento projetado para encapsular uma grande variedade de pacotes da camada de rede do protocolo IP. O GRE foi desenvolvido pela Cisco e projetado para ser stateless. Comparado a outros protocolos de tunelamento, como o IPSec, o GRE possui a habilidade de trabalhar com protocolos multicast. Um exemplo é a utilização do OSPF em um túnel GRE.

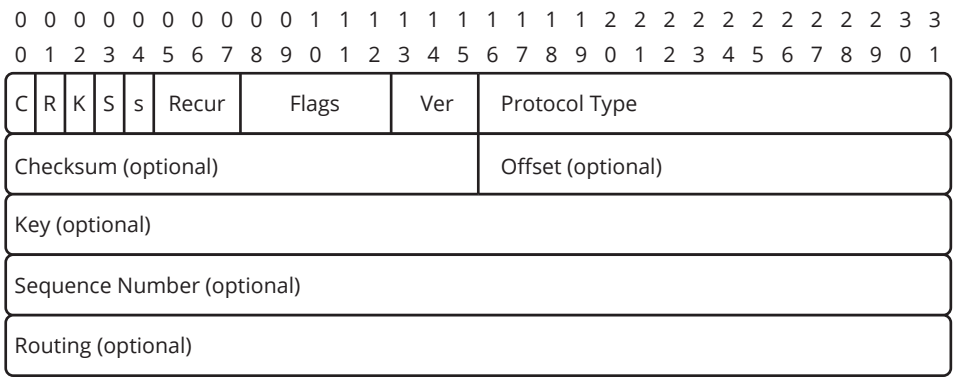

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
C
R
K
S
s
Recur
Flags
Ver
Protocol Type
Checksum (optional)
Offset (optional)
Key (optional)
Sequence Number (optional)
Routing (optional)

Figura 9.5 Protocolo Generic Routing Encapsulation (GRE).

Túneis GRE são normalmente configurados nos roteadores de borda (configuração ponto-a-ponto). Os pacotes a serem enviados através do túnel são encapsulados por um novo cabeçalho (cabeçalho GRE) e colocados no túnel com o endereço de destino do final do túnel. Ao chegar a esse final, os pacotes são desencapsulados (retira-se o cabeçalho GRE) e continuarão seu caminho para o destino determinado pelo cabeçalho original.

Desvantagens:

- Os túneis GRE são geralmente configurados manualmente;
- Quanto maior a quantidade de túneis, maior será o consumo da banda de rede e do processamento dos roteadores.

Na figura anterior pode ser observado o datagrama do protocolo GRE. Explicação de cada campo:

- **C, Checksum Presente (1 bit)**: ativa a verificação do campo "Checksum".
- **R, Routing Presente (1 bit)**: ativa a verificação do campo "Routing".
- **K, Key Presente (1 bit)**: informa que o campo "Key" possui informações válidas.
- **S, Number Sequence Presente (1 bit)**: informa que o campo "Sequence Number" possui informações válidas.
- **s, Strict Source Route (rota de origem restrita – 1 bit)**: bit reservado para implementações futuras.
- **Recur – Recursion Control (controle de reincidência – 3 bits)**: contém o número adicional de encapsulamentos permitidos; 0 é o padrão.
- **Flags (5 bits)**: bits reservados que devem ser transmitidos como 0.
- **Ver (versão – 3 bits)**: versão do protocolo GRE; deve ser setado para 0.
- **Type Protocol (tipo do protocolo – 16 bits)**: contém o tipo do protocolo utilizado pelo payload. Em geral, o valor será o definido para o protocolo Ethernet. Valores adicionais podem ser definidos em outros documentos.
- **Checksum (16 bits; opcional)**: código utilizado para verificar erros de transmissão.
- **Offset (16 bits; opcional)**: indica o byte offset de início do campo Routing e o primeiro byte da entrada ativa do roteamento de origem para ser examinado.
- **Key (chave – 32 bits)**: contém a chave definida pelo encapsulator, que pode ser usada pelo receptor para autenticar a fonte dos dados.
- **Sequence Number (número sequencial)**: contém um número inserido pelo encapsulator, que pode ser usado pelo receptor para estabelecer a ordem em que os pacotes foram transmitidos.
- **Routing (roteamento)**: esse campo é uma lista de SREs.

Protocolo PPTP

O Point-to-Point Tunneling Protocol (PPTP) foi desenvolvido como uma extensão do protocolo ponto-a-ponto (PPP). Ele encapsula os protocolos IP, NetBEUI ou IPX em datagramas do PPP dentro de um cabeçalho GRE. Na prática, por se tratar de um protocolo baseado na camada 2 do modelo OSI (enlace), você pode executar remotamente aplicativos que dependem de determinados protocolos de rede privada. O servidor de encapsulamento efetua todas as verificações e validações de segurança, além de permitir a criptografia de dados, o que torna mais seguro o envio de informações em redes que não sejam seguras. O protocolo PPTP também pode ser utilizado entre redes locais privadas.

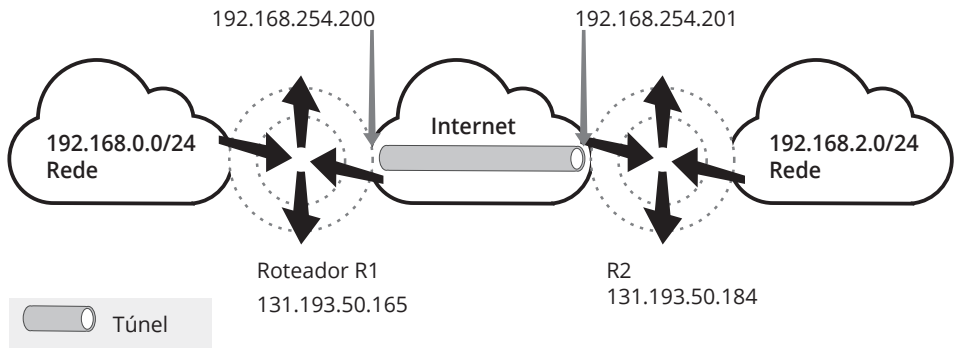


Figura 9.6
O Point-to-Point Tunneling Protocol.

PPTP é um protocolo proprietário da Microsoft que tem como principal desvantagem o fato de não implementar o uso de criptografia de forma nativa.

Protocolo L2F

- Protocolo de camada 2 criado pela Cisco.
- Permite o tunelamento de um link de comunicação encapsulando o PPP/SLIP.

O protocolo Layer-2 Forwarding (L2F) é usado para estabelecer um túnel seguro em uma infraestrutura pública (como a internet) que liga um POP ISP a um gateway interno de uma empresa. Isso cria um túnel virtual ponto-a-ponto entre o usuário e a rede interna da empresa em que o cliente trabalha.

Por ser um protocolo de camada 2, o L2F permite o tunelamento de um link de comunicação (exemplo: HDLC, async HDLC ou frames SLIP) encapsulando o PPP/SLIP dentro de um pacote L2F. O NAS ISP e o home gateway exigem um entendimento comum de modo a permitir a transmissão e recepção com sucesso em toda a internet dos pacotes encapsulados SLIP/PPP.

As principais funções do protocolo Cisco L2F foram cobertas pelo protocolo L2TP, atualmente o protocolo padrão utilizado para tunelamento.

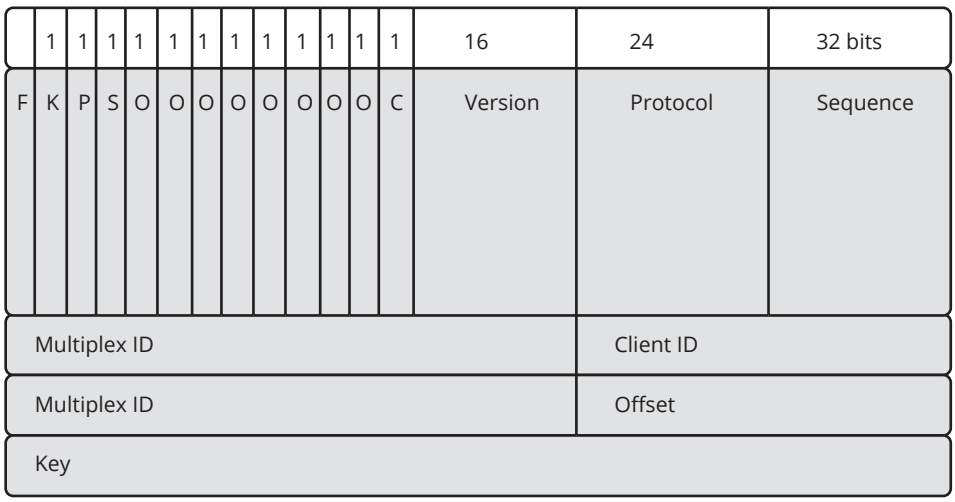


Figura 9.7 Protocolo L2F (Layer-2 Forwarding).

A estrutura básica do protocolo pode ser observada na Figura 9.7. Principais atributos do cabeçalho:

- **Version**: versão do software que criou o pacote;
- **Protocol**: determina o tipo do protocolo utilizado para transportar os pacotes L2F;
- **Sequence**: o número de sequência está presente se o bit S estiver configurado para 1 no cabeçalho L2F;
- **Multiplex ID**: o pacote multiplex ID identifica um tipo particular de conexão no interior do túnel;
- **Client ID**: o Cliente ID auxilia os endpoints nos túneis de multiplexing;
- **Length**: determina o tamanho total do pacote em octetos, incluindo o cabeçalho, todos os campos e o payload;
- **Offset**: esse campo identifica o número de bytes antes do cabeçalho L2F e após o início do payload de dados. Esse campo está presente se o bit F estiver setado no cabeçalho L2F;

- **Key**: o campo chave está presente se o bit K estiver setado no cabeçalho L2F; faz parte do processo de autenticação;
- **Checksum**: o checksum do pacote. O campo checksum está presente se o o bit C estiver configurado no cabeçalho L2F.

Protocolo L2TP

O Layer 2 Tunnel Protocol (L2TP) é um protocolo definido pelo Internet Engineering Task Force (IETF), que combina as características de dois protocolos existentes no mercado: Cisco L2F (Layer 2 Forwarding) e Microsoft PPTP (Point-to-Point Tunneling Protocol). L2TP é uma extensão do protocolo Point-to-Point Protocol (PPP), que é um importante componente das VPNs.

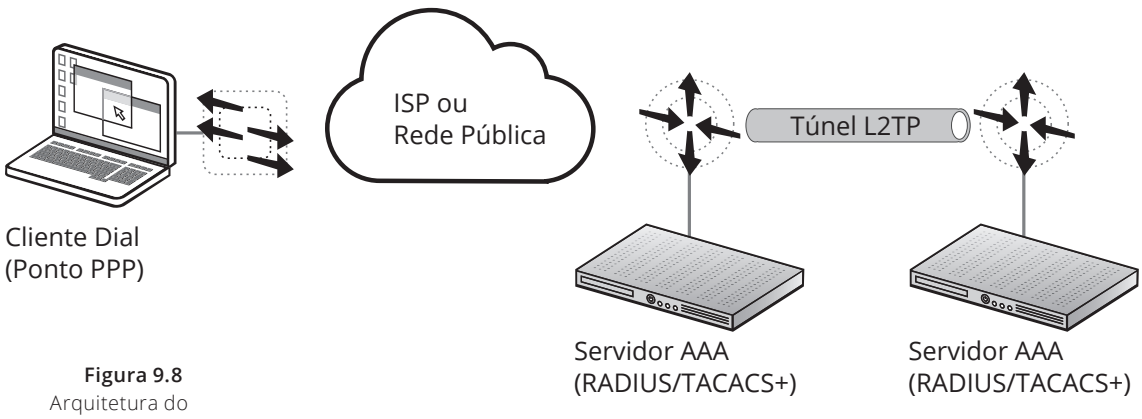


Figura 9.8
Arquitetura do Protocolo L2TP.

Na imagem, temos os seguintes objetos envolvidos no protocolo L2TP:

- **Cliente Dial**: sistema final ou roteador que origina uma sessão PPP e que está conectado a uma RTPC ou RDSI, podendo ser o iniciador ou o terminador da chamada.
- **L2TP Access Concentrator (LAC)**: dispositivo L2TP ao qual o cliente conecta diretamente e através do qual quadros PPP são "tunelados" para o L2TP Network Server (LNS); qualquer protocolo transportado pelo PPP pode ser tunelado; o LAC é o iniciador das chamadas de entrada e o recebedor das chamadas de saída; análogo ao NAS.
- **L2TP Network Server (LNS)**: ponto de terminação do túnel L2TP e ponto de acesso onde os quadros PPP são processados e entregues para os protocolos das camadas superiores; o LNS é o lado servidor do protocolo L2TP, suportando interfaces LAN ou WAN e podendo terminar chamadas de qualquer interface PPP; o LNS é o iniciador das chamadas de saída e o recebedor das chamadas de entrada; análogo ao HGW.
- **Servidor AAA**: servidor de autenticação de clientes que usa os protocolos Radius ou TACACS+; fornece o serviço AAA (Authentication, Authorization, Accounting).

Tunelamento L2TP

Usando L2TP, um provedor de serviços internet (ISP) pode criar um túnel virtual que conecta sites remotos de clientes ou usuários remotos às redes corporativas. O L2TP Access Concentrator (LAC) localizado nos pontos de presença (POP) do ISP troca mensagens PPP com os usuários remotos e se comunica via protocolo L2TP com o L2TP Network Server (LNS) para estabelecer os túneis.

O L2TP encaminha os pacotes através do túnel virtual criado entre os pontos finais da conexão ponto-a-ponto. Os quadros originados pelos usuários remotos são aceitos no POP

do ISP; são retirados os bytes de controle do quadro, encapsulados no L2TP e enviados pelo túnel. O home gateway do cliente aceita os quadros L2TP, retira o encapsulamento L2TP e processa os quadros para a interface adequada.

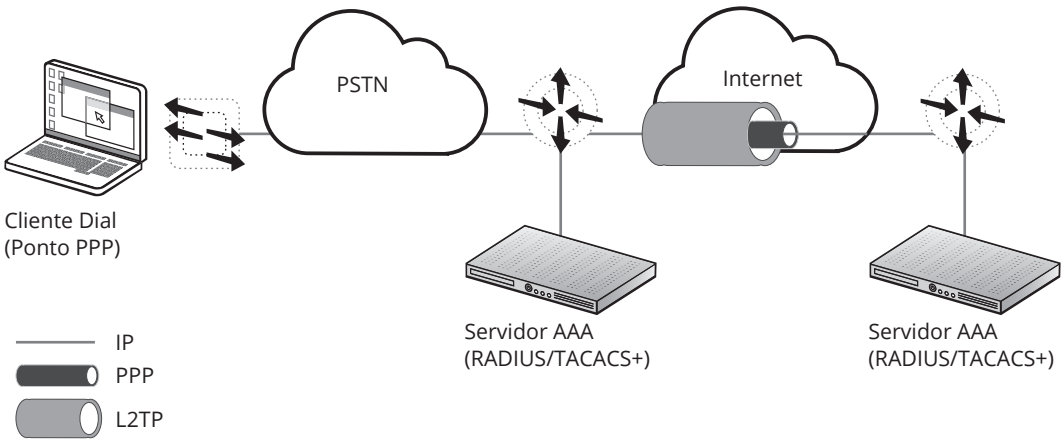


Figura 9.9
Estrutura e encapsulamento do túnel L2TP.

Na imagem vemos que os quadros PPP (transportando pacotes IP) originados no usuário passam pela RTPC sem proteção (supõe-se que o nível de segurança da RTPC é adequado, além de ser certamente maior do que o nível de segurança da internet) e são "tunelados" no trecho entre o POP do ISP (onde fica o LAC) até a rede corporativa (onde fica o LNS), que é o trecho dentro da internet onde o nível de segurança tem de ser maior.

Protocolo IPSec

- Os protocolos PPTP, L2F e L2TP não incluem recursos de criptografia ou processamento para chaves criptográficas.
- IPSec (IP Security) foi criado para suprir tais necessidades.

Os protocolos PPTP, L2F e L2TP não incluem criptografia ou processamento para tratar chaves criptográficas, o que é bastante recomendado para garantir a segurança dos pacotes. Por isso, surgiu um dos mais importantes protocolos, criado para garantir a segurança da próxima geração de pacotes IP (IPv6) e que, no momento, vem sendo utilizado com protocolos IPv4. O IPSec permite ao usuário ou ao gateway seguro que está agindo em seu favor autentificar ou criptografar cada pacote IP, ou ainda fazer os dois processos simultaneamente.

VLAN

- A Virtual LAN (VLAN) resolve problemas de colisão e broadcast em redes internas com muitos hosts.
- Isso ocorre através do desenvolvimento de switchs que permitam dividir uma rede em vários segmentos.
- O protocolo utilizado normalmente é o IEEE 802.1q.
- As VLANs podem ser configuradas com base no protocolo, no endereço MAC, no endereço IP, na porta do switch ou na marcação e filtragem do quadro (tagging).
- Implementação: Estática x Dinâmica.

As VLANs surgiram para resolver problemas relacionados ao constante crescimento das redes internas. Nesse cenário, a grande quantidade de equipamentos (hosts) em um mesmo segmento de rede tende a provocar colisões constantes e perda de tempo processando grandes quantidades de pacotes de broadcast (mesmo domínio de broadcast). A solução para esse problema é desenvolver comutadores (switchs) que permitam criar redes virtuais independentes que operem na camada 2 do modelo OSI. Essa implementação permite dividir uma rede em segmentos menores diminuindo, por consequência, a quantidade de colisões e broadcast. No entanto, uma VLAN geralmente é configurada para mapear diretamente uma rede ou subrede IP, o que dá a impressão de que a camada 3 está envolvida.

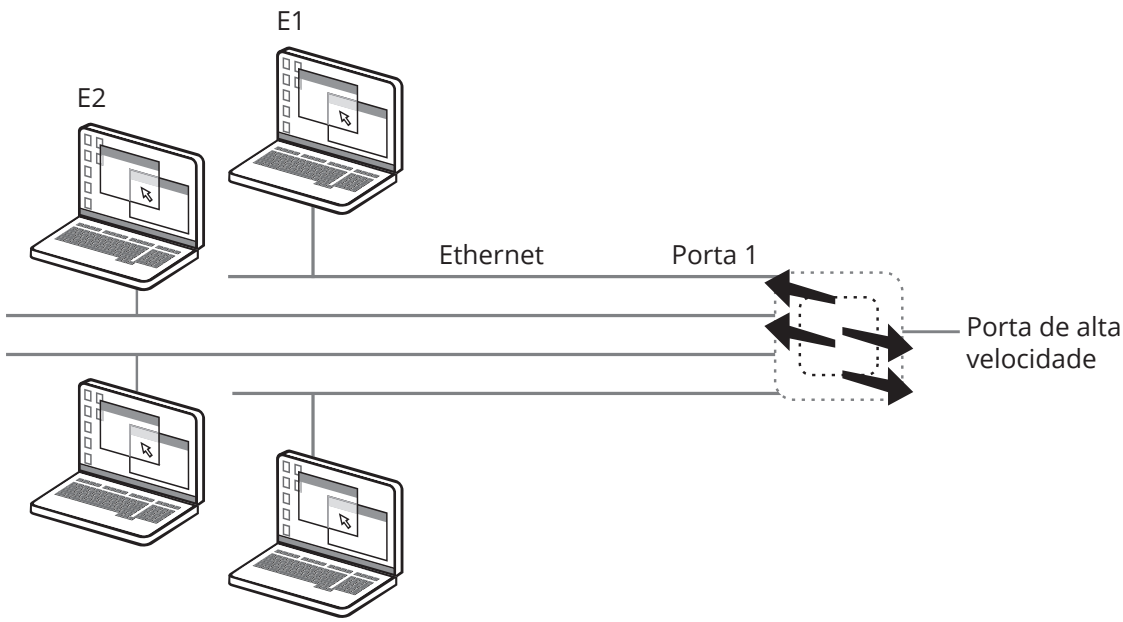
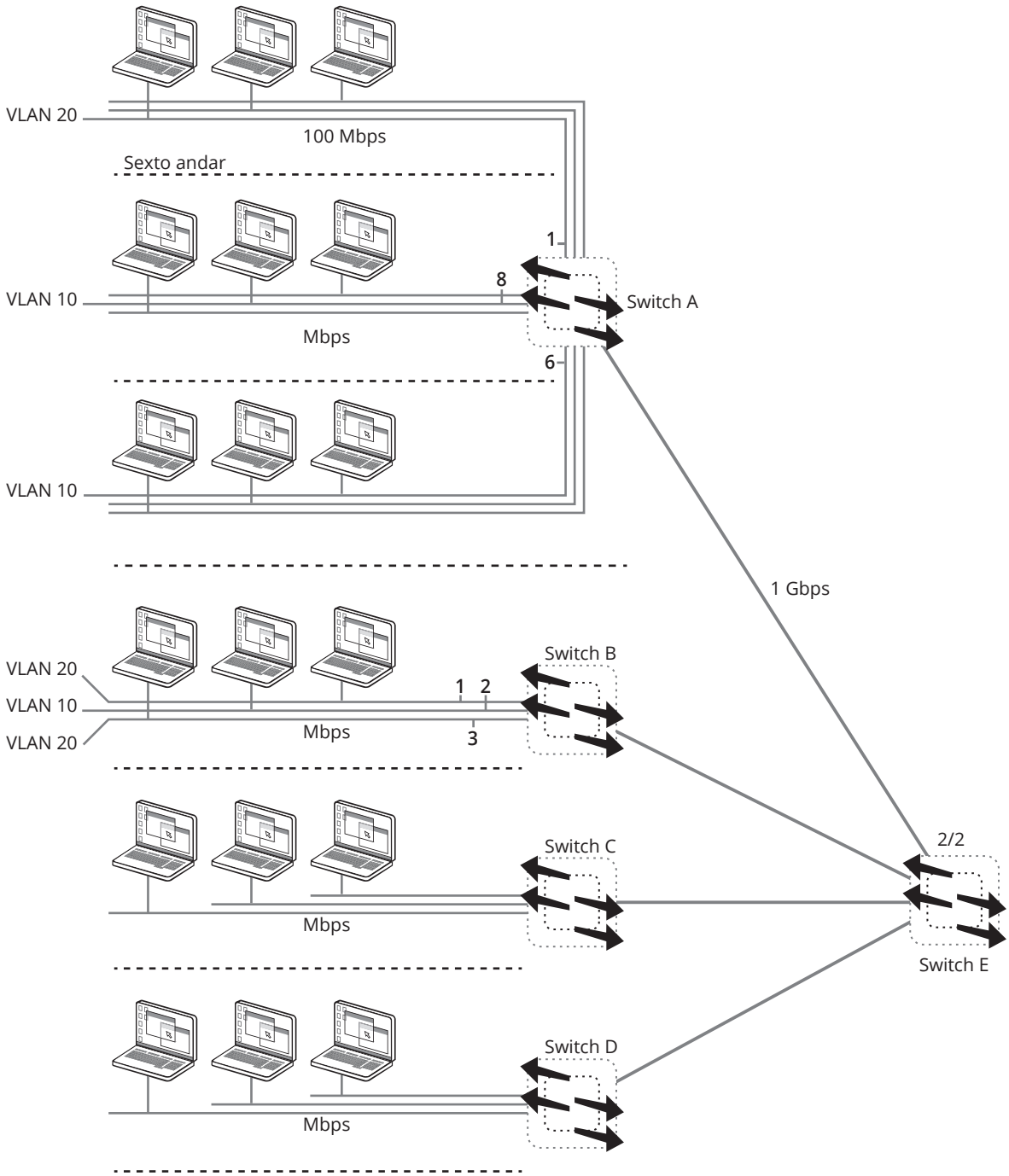


Figura 9.10
VLAN (Virtual LAN).



Uma rede local virtual (VLAN) é uma rede logicamente independente com a possibilidade de coexistir em um mesmo comutador (switch).

As VLANS podem ser configuradas de várias formas:

- Nível do protocolo, IP, IPX, LAT etc.
- Baseada no endereço MAC.
- Baseada na sub-rede IP.

Figura 9.11
Switchs: solução para casos de grande quantidade de equipamentos (hosts) em um mesmo segmento de rede.

- Baseada na porta do switch, ou seja, você pode representar as VLANs informando que as portas 1, 2, 5 e 8 pertencem ao segmento (VLAN) A, e 3, 6 e 9 pertencem ao segmento (VLAN) B. Essa representação possui uma relação direta com o mundo real, pois você pode separar a empresa por departamentos. Por exemplo: departamento de finanças do departamento de recursos humanos.
- Baseada em marcação de quadro (frame-tagging) e filtragem de quadro (frame-filtering). A marcação de quadro modifica a informação contida dentro do quadro da camada 2, de modo que os switchs possam encaminhar o tráfego da VLAN para as suas VLANs de destino e voltar o quadro ao seu formato normal.

O protocolo utilizado para gerência de VLAN é, atualmente, o IEEE 802.1Q. Esse protocolo introduz a técnica conhecida como tagging e o conceito de VLAN nativa.

As VLANs podem ser Estáticas ou Dinâmicas:

- **VLAN Estáticas**: baseadas na configuração das portas, ou seja, você informa que qualquer dispositivo conectado a uma determinada porta do comutador pertence a uma determinada VLAN.
- **VLAN Dinâmicas**: baseadas em endereços MAC, endereço IP, tipo de protocolo ou frame-tagging tag. O administrador da rede deve previamente cadastrar os endereços nos parâmetros a serem utilizados pelo comutador e associá-los às suas respectivas VLANs. Com isso, quando o usuário plugar seu micro na rede, independente da porta, ele será alocado para a VLAN correta.

Qualidade de Serviço (QoS) – MPLS

- O objetivo inicial do MPLS foi melhorar a velocidade de encaminhamento dos pacotes pela rede IP.
- Em resumo, o MPLS gera uma etiqueta de comprimento fixo reduzido que atua como uma forma abreviada do cabeçalho IP.
- Somente os roteadores de borda analisam o cabeçalho IP.
- Possibilita estabelecer QoS (Quality of Services) em redes IP.
- Possibilita VPNs multimídia.
- Protocolo padronizado pelo IETF.

O Multiprotocol Label Switching (MPLS) foi apresentado, inicialmente, como uma solução que possibilita melhorar a velocidade de encaminhamento dos pacotes por uma rede IP, mas agora está sendo considerado uma tecnologia de grande importância, que oferece novas potencialidades. Duas aplicações principais para a tecnologia MPLS estão ligadas à engenharia de tráfego e suporte à VPN.

A essência do MPLS é a geração de uma etiqueta de comprimento fixo reduzido que atua como uma forma abreviada do cabeçalho IP. Esse mesmo conceito é utilizado nos correios, onde a rua e a cidade são representados por um endereço postal, de tamanho fixo reduzido, que é o CEP. Através do uso dessas etiquetas criadas no MPLS são tomadas as decisões no encaminhamento do pacote. Os pacotes IP possuem um campo em seu cabeçalho que contém o endereço para o qual o pacote será encaminhado. No roteamento tradicional, esse campo é verificado e processado em cada roteador da rede até que ele atinja seu destino (roteamento hop by hop).

No MPLS, os pacotes IP são encapsulados, através do uso de etiquetas, pelos dispositivos que se encontram na entrada da rede (roteador de borda). O roteador de borda do MPLS analisa os índices do cabeçalho IP e seleciona uma etiqueta apropriada com que o pacote será encapsulado. Grande parte do poder do MPLS vem do fato de que, em contraste

ao roteamento tradicional IP, esta análise pode ser baseada não apenas no endereço de destino que ele carrega dentro do cabeçalho, mas também pelo QoS (Qualidade de Serviço) requerido. Em todos os nós subsequentes dentro da rede MPLS, a etiqueta é utilizada pelos roteadores para realizar a decisão de encaminhamento dos pacotes na rede. Em nenhum momento os roteadores pertencentes ao núcleo da rede analisam o cabeçalho IP. Finalmente, à medida que os pacotes deixam a rede, as etiquetas são retiradas pelos roteadores de borda da rede.

Na terminologia do MPLS, os roteadores são chamados de Roteadores de Comutação por Etiquetas (Label Switched Routers – LSR). Os roteadores MPLS encaminham os pacotes tomando as decisões de encaminhamento com base na etiqueta MPLS.

Os roteadores IP convencionais contêm tabelas de roteamento onde são feitas buscas referentes à informação do cabeçalho IP de um pacote para que a decisão de encaminhamento seja tomada. Estas tabelas são construídas pelos protocolos de distribuição do IP (por exemplo, RIP ou OSPF). O roteamento em geral engloba o plano de encaminhamento e o plano de controle. No roteamento IP, a análise do cabeçalho é feita no plano de encaminhamento, e no plano de controle é gerada a tabela de roteamento. No MPLS, é possível separar o plano de encaminhamento do plano de controle. Com isso, é possível modificar cada um separadamente.

Devido à essa característica, não é necessário, por exemplo, mudar os dispositivos de encaminhamento caso se deseje mudar a estratégia de roteamento da rede.

Existem duas categorias de roteadores no MPLS. Na borda da rede, os classificadores de pacote precisam de um desempenho elevado no processo de aplicação (e retirada) das etiquetas: eles são conhecidos como roteadores de borda (Edge Label Switching Label – ELSR). A outra classe é composta pelos roteadores de núcleo. Uma de suas características principais é encaminhar os pacotes etiquetados de forma rápida. Por isso há necessidade de um grau de processamento elevado.

Diversos conceitos básicos que se aplicam à tecnologia de comutação precisam ser revistos antes de descrever como o MPLS trabalha.

Roteamento

É um termo usado para descrever as ações feitas pela rede para mover os pacotes através dela. Nos referimos a pacotes roteados de a para b: ou estão sendo roteados em uma rede ou em uma rede interna. Os pacotes seguem através da rede, sendo enviados de uma máquina a outra até atingir seu destino. O protocolo de roteamento (por exemplo: RIP, OSPF) capacita cada máquina a compreender a outra máquina, que é o próximo “hop” que um pacote deve fazer para atingir seu destino. Os roteadores usam os protocolos de roteamento para construir as tabelas de roteamento. Quando recebem um pacote e têm de tomar uma decisão de encaminhamento, os roteadores procuram na tabela de roteamento usando o endereço IP do destino no pacote, obtendo desse modo a identidade da máquina pertencente ao próximo “hop”. A construção das tabelas e seu uso para o encaminhamento podem ser separados por operações lógicas. A figura mostra essas funções, à medida que ocorrem no roteador.

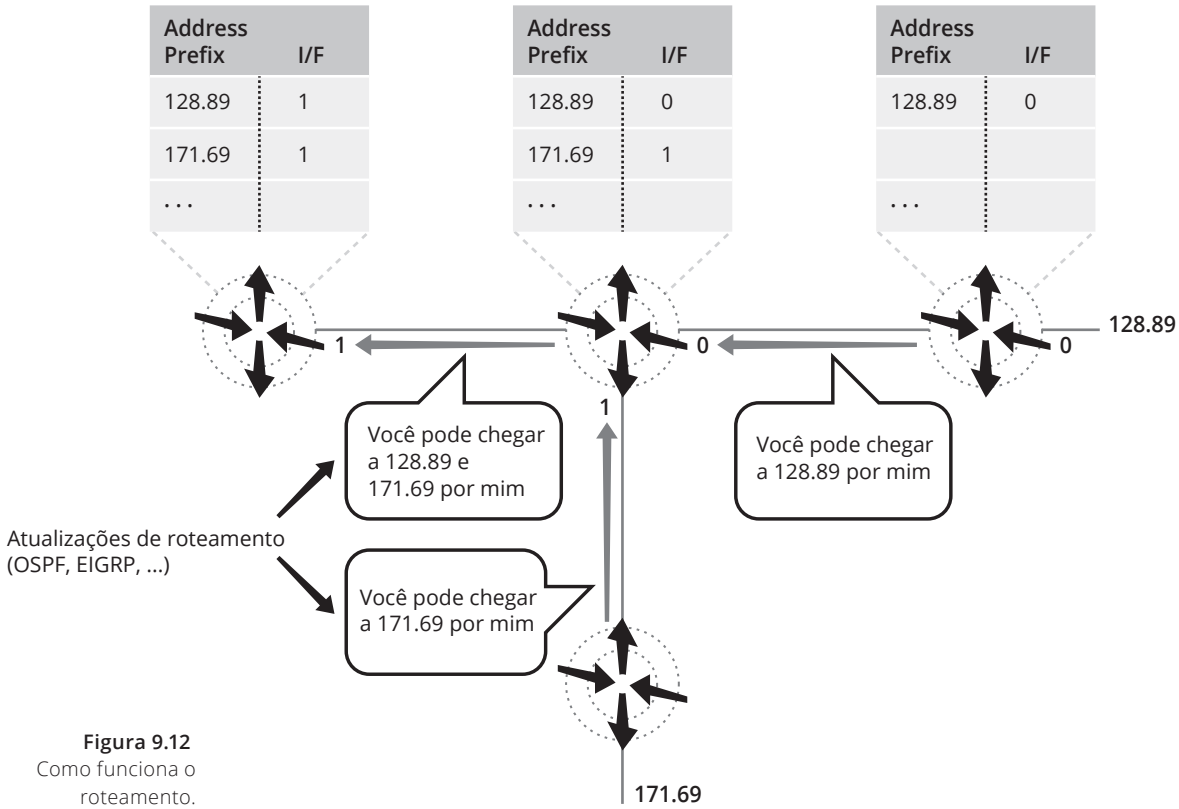


Figura 9.12
Como funciona o roteamento.

Encaminhamento dos pacotes

A figura a seguir ilustra o encaminhamento de um pacote baseado apenas no endereço IP de destino, depois que os roteadores "aprenderam" as rotas através da troca de informações entre os protocolos de roteamento.

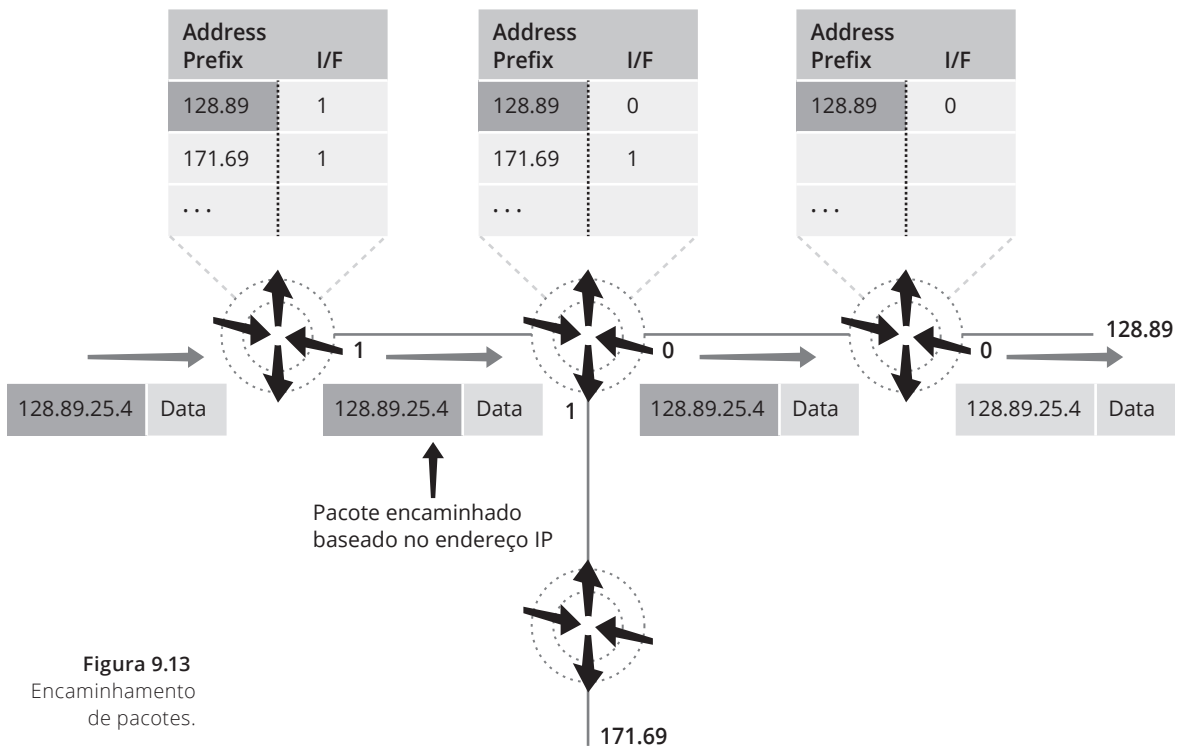


Figura 9.13
Encaminhamento de pacotes.

Comutação

- Transferência de dados de uma porta de entrada para uma porta de saída.

Componente de controle:

- Constrói e mantém em uso a tabela de encaminhamento do nó.

Componentes de encaminhamento:

- Executam o encaminhamento dos pacotes.

Processo geralmente usado para descrever a transferência de dados de uma porta de entrada para uma porta de saída, onde a escolha da porta de saída é baseada em informação da camada 2 (por exemplo, ATM VPI/VCI).

Componente de controle

Constrói e mantém em uso a tabela de encaminhamento do nó. O roteador trabalha com componentes de controle de outros nós para distribuir informação de roteamento de forma consistente e precisa, e também assegura procedimentos locais usados nas tabelas de encaminhamento. Protocolos de roteamento padrão (por exemplo OSPF, BGP e RIP) são usados na troca de informações de roteamento entre os componentes de controle. O componente de controle precisa reagir quando ocorrem mudanças na rede (como a falha de um link), mas não é envolvido no processamento individual dos pacotes.

Componentes de encaminhamento

Executam o encaminhamento dos pacotes. Eles usam as informações a partir da tabela de encaminhamento (mantido pelo roteador); essa informação é carregada no próprio pacote; um conjunto de procedimentos locais faz a decisão de encaminhamento. No roteamento convencional, um algoritmo compara o endereço de destino no pacote com uma entrada na tabela de roteamento até que seja encontrado um valor vantajoso. Todo esse processo é repetido em cada nó da origem até ao destino. Nos LSR, algoritmos de troca de etiquetas usam as etiquetas dos pacotes e a tabela de encaminhamento baseada em etiquetas para obter uma nova etiqueta e uma interface de saída para o pacote.

Label Switching

As etiquetas são colocadas pelo LSR, que enviará o pacote. O LSR que recebe esse pacote etiquetado precisa saber o que fazer com ele. É de responsabilidade do componente de controle assumir essa tarefa. Ele usa como guia o conteúdo de entrada da tabela de encaminhamento.

O estabelecimento e a manutenção da entrada da tabela de encaminhamento são funções essenciais desenvolvidas pelo LSR. O componente de controle é responsável por distribuir a informação de forma consistente entre os LSRs e executar os procedimentos usados pelos mesmos para converter essas informações para uma tabela de encaminhamento.

	Roteamento Convencional	Comutação de Etiqueta
Análise do cabeçalho IP por completo	Ocorre em todo nó.	Ocorre somente na borda da rede quando a etiqueta é determinada.
Suporte unicast e multicast	Necessita de muitos algoritmos complexos de encaminhamento.	Necessita de um algoritmo de encaminhamento.
Decisão de roteamento	Baseia-se somente no endereço.	Inúmeros parâmetros podem ser usados como base, tais como: QoS, VPN, membership.

Tabela 9.2
Convencional e Comutação de Etiqueta.

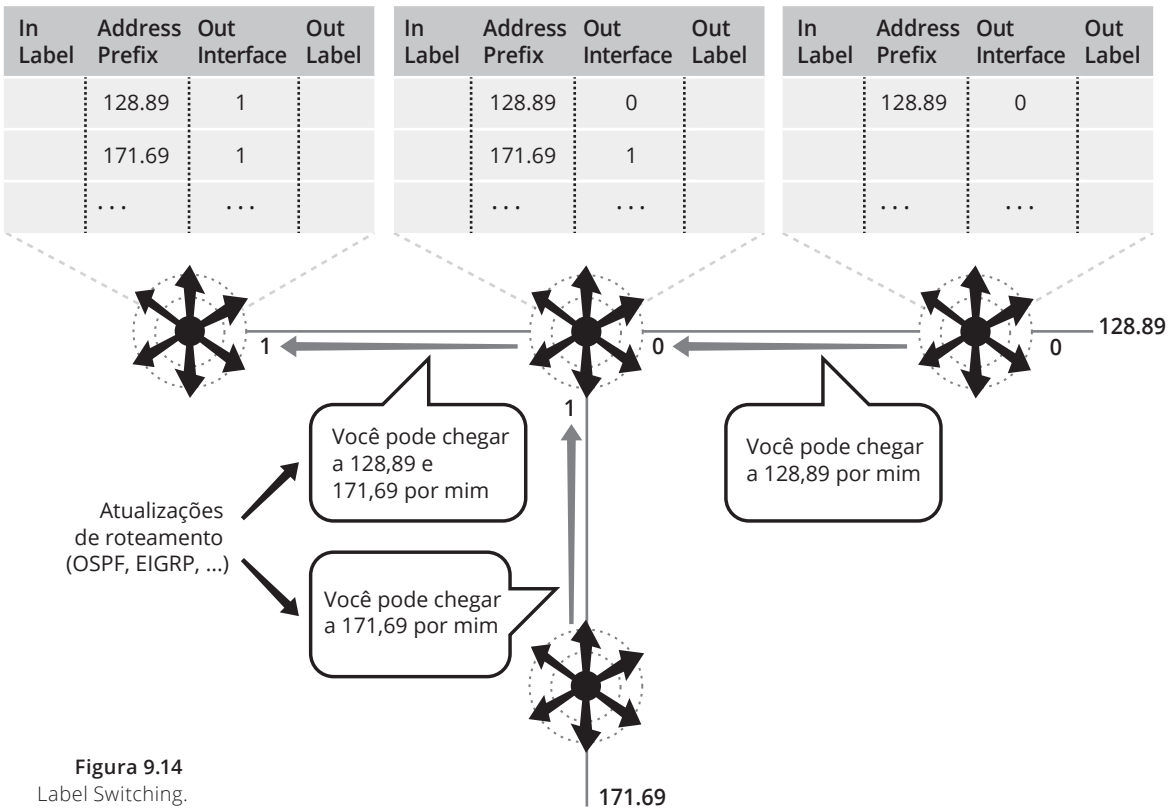


Figura 9.14
Label Switching.

Os componentes de controle da comutação de etiquetas incluem todos os protocolos convencionais de distribuição (por exemplo: OSPF, BGP, PIM e assim por diante). Esses protocolos de distribuição fornecem aos LSRs informações sobre como fazer o mapeamento entre o FEC e os endereços dos equipamentos referentes ao próximo salto. Além disso, o LSR deve:

- Criar a ligação entre as etiquetas e os FECs;
- Distribuir estas ligações para outros LSRs;
- Construir sua própria tabela de encaminhamento.

A ligação entre uma etiqueta e um FEC pode ser orientada a dados (resultado da presença de tipos específicos de fluxo de tráfego) ou orientada a controle (orientação pela topologia referente às atualizações de roteamento ou outras mensagens de controle).

Cada uma dessas técnicas de ligação possui numerosas opções. A decisão para estabelecer o fluxo pode ser baseada em vários critérios. A ligação da etiqueta, no caso de orientação a dados, estabelece etiquetas ativas somente quando existe uma necessidade imediata (isto é, quando o tráfego é apresentado para o encaminhamento). Informações sobre mudança de topologia ou mudança de tráfego precisam ser distribuídas.

> No caso da orientação a controle, a ligação é baseada no conhecimento resultante do procedimento de roteamento e da reserva de recursos.

Informação de distribuição das etiquetas

A entrada de uma tabela de encaminhamento fornece, no mínimo, informações sobre a interface de saída e a nova etiqueta, mas também contém outras informações. Pode, por exemplo, indicar o método de enfileiramento na saída a ser aplicado ao pacote.

Cada etiqueta distribuída deve ser limitada a uma entrada na tabela do encaminhamento. Essa ligação pode ser executada no LSR local ou ser fornecida por um LSR remoto.

A arquitetura MPLS usa um controle local (o LSR pode criar e anunciar uma ligação sem esperar uma comunicação do vizinho sobre o mesmo FEC) e controle de saída (o LSR espera uma comunicação de seu vizinho antes de alocar uma etiqueta).

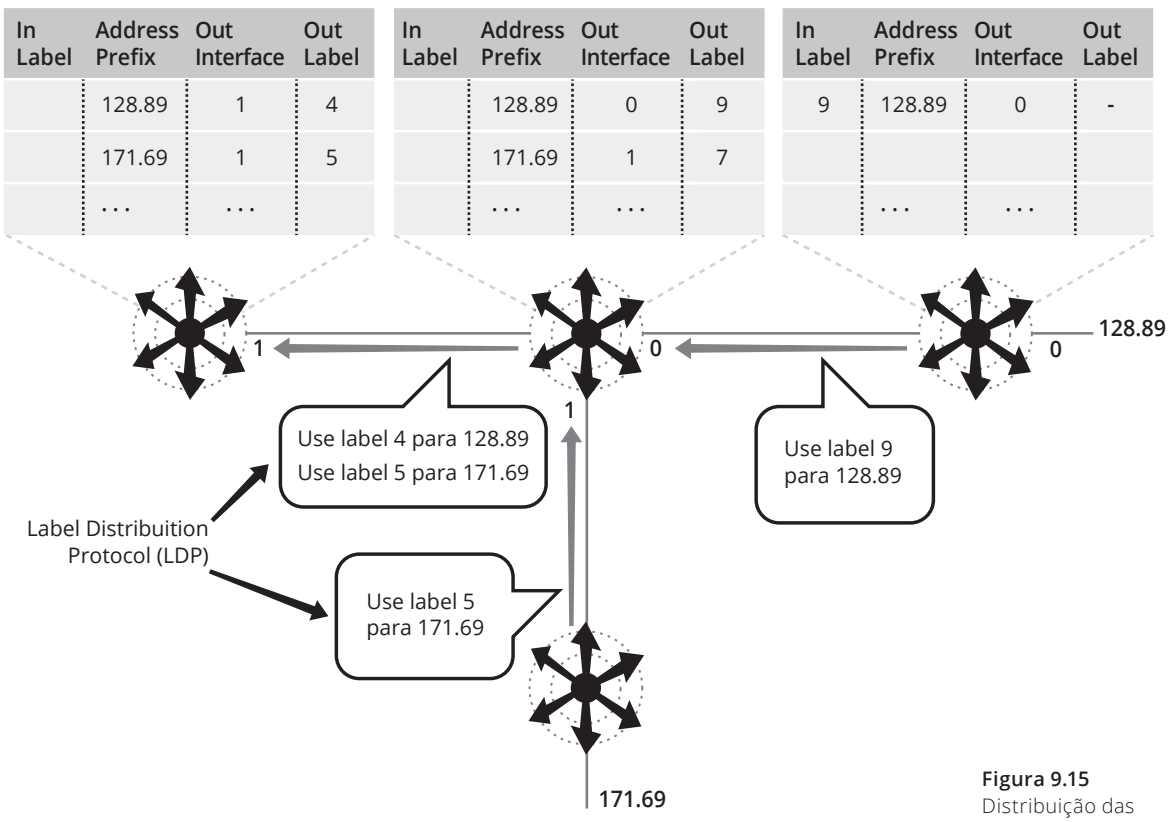


Figura 9.15
Distribuição das etiquetas.

O conhecimento entre as ligações localmente escolhidas e os FECs associados deve ser disseminado aos LSRs adjacentes para que essas informações sejam usadas na construção das tabelas de encaminhamento. A informação na tabela de encaminhamento deve seguir também as mudanças na rede. Depois de tudo, a etiqueta no pacote de entrada é usada para descobrir as regras para o encaminhamento do pacote.

A informação da etiqueta pode ser distribuída de duas maneiras:

- **Adicionando nos protocolos de roteamento**: as informações podem ser adicionadas nos protocolos tradicionais de roteamento, embora somente os esquemas orientados a controle possam suportar esse método. A troca em operações normais dos protocolos assegura a consistência na informação de encaminhamento e evita a necessidade de outros protocolos.
- **Uso do protocolo de distribuição de etiqueta**: seguindo o modelo Tag Switching da Cisco, o grupo que trabalha na definição do MPLS criou um novo protocolo específico para a distribuição de etiquetas, chamado Protocolo de Distribuição de Etiquetas (Label Distribution Protocol – LDP), que pode ser usado nos esquemas de orientação a dados e também de controle. A desvantagem do LDP é que ele é mais complexo.

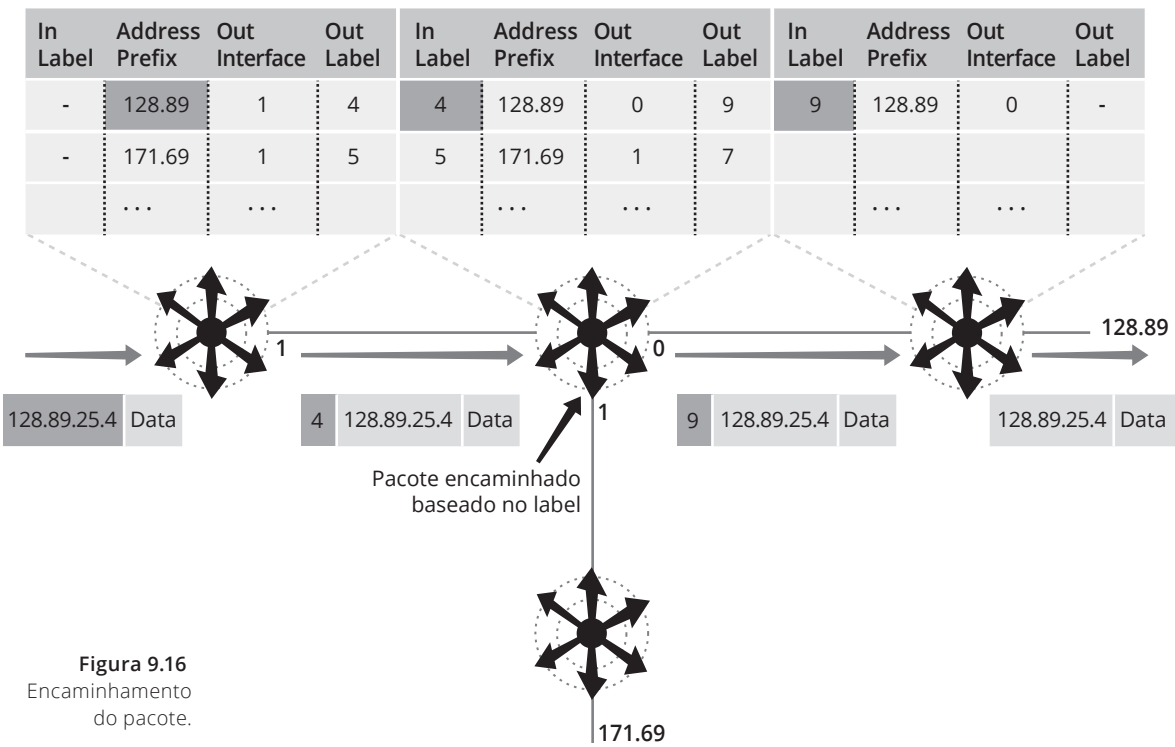


Figura 9.16
Encaminhamento do pacote.

Roteador de Borda

- Edge Label Switching Router (ELSR).
- Responsável por classificar o tráfego e aplicar e remover etiquetas dos pacotes.
- Determina se o tráfego é um fluxo estável.
- Implementa políticas de gerência e controle de acesso.

É de responsabilidade dos Edge Label Switching Router ou Roteadores de Borda (ELSRs) classificar o tráfego e aplicar ou remover (entrada/saída) etiquetas aos pacotes. Como foi visto anteriormente, as etiquetas podem ser atribuídas tomando como base requisitos de QoS e não o seu endereço de destino, como é feito no roteamento tradicional. O ELSR determina se o tráfego é um fluxo estável e implementa políticas de gerência e controle de acesso.

Assim, a capacidade dos ELSRs é a chave para o sucesso de um ambiente de comutação de etiquetas. É também um ponto do controle e gerência dos provedores do serviço.

Novas gerações de ELSR precisam ter as seguintes capacidades:

- **Potencialidade de classificação do fluxo IP**: permitirá que esses equipamentos atribuam valores de QoS e apliquem etiquetas ao IP sem nenhuma degradação no desempenho do encaminhamento;
- **Potencialidade extensiva a VPN**: esses equipamentos precisam rodar múltiplas tabelas de roteamento para que os clientes das VPN possam separar seu tráfego.
- Roteamento explícito.
- Redes privadas virtuais.
- Suporte a múltiplos protocolos e a múltiplos links.
- Facilidade de evolução.
- Roteamento inter-domínio.
- Suporte a todos os tipos de tráfego.

Vantagens do MPLS

Uma das principais vantagens do MPLS é que ele é a base do padrão para a tecnologia de comutação por etiqueta. O desenvolvimento de um padrão resulta em um ambiente aberto com vários fabricantes produzindo equipamentos interoperáveis. A competição também resulta em preços mais baixos e conduz a uma maior inovação.

Roteamento explícito

O roteamento explícito é uma técnica poderosa, podendo ser aplicada para vários propósitos. O roteamento implícito baseado na análise dos datagramas (pacote a pacote) para muitas aplicações gera uma sobrecarga muitas vezes inaceitável. O MPLS permite que os pacotes sejam classificados a partir de etiquetas atribuídas na admissão dos nós MPLS, e encaminhados, dentro de uma mesma classe, num caminho virtual, sem a necessidade de ser analisado nó a nó. O roteamento explícito tem também a vantagem de criar "túneis transparentes" por onde passa qualquer tipo de tráfego (por exemplo: SNA e IPX). Os LSRs "enxergam" apenas as etiquetas dos pacotes enviados pelo túnel.

Redes Privadas Virtuais (Virtual Private Netwoks – VPN)

Muitas empresas constroem redes privadas para conectar vários locais. O objetivo é ter uma rede de transporte que ofereça segurança, confiança, comportamento previsível e que seja mais barata. VPN é uma emulação dessa rede privada. MPLS é um ingrediente -have na construção dessas redes; as etiquetas do MPLS podem ser usadas para isolar o tráfego entre VPNs.

Suporte a múltiplos protocolos e a múltiplos links

O componente de encaminhamento não é específico a uma camada da rede. Por exemplo, o mesmo componente de encaminhamento poderia ser usado para fazer a comutação de etiqueta no IP assim como a comutação de etiqueta no IPX. A comutação por etiquetas pode operar sobre todos os protocolos da camada de enlace de dados, embora a ênfase inicial estivesse no ATM.

Facilidade de evolução

A comutação de etiquetas tem a vantagem de prover uma separação entre as funções de controle e encaminhamento. Cada parte pode evoluir sem impactar a outra parte, o que faz a evolução da rede ser mais fácil, de menor custo e menos propensa a erros.

Roteamento Inter-Domínio

A comutação de etiquetas fornece uma separação mais completa entre a distribuição intra-domínio e interdomínio. Isso melhora a escalabilidade no processo de roteamento e, de fato, reduz o conhecimento requerido de uma rota dentro de um domínio. Esse é um benefício aos ISPs (Internet Service Providers) e aos portadores que podem ter uma quantidade grande de tráfego em trânsito, isto é, o tráfego cuja fonte e destino não estão na mesma rede.

Suporte a todos os tipos de tráfego

Outra vantagem da comutação de etiqueta que geralmente não é visível ao usuário é o suporte a todos os tipos de encaminhamento: unicast, unicast com tipo de serviço e pacotes multicast. O suporte pode ser usado com atributos de QoS que, por sua vez, permitem que sejam definidas diferentes classes de serviços de acesso aos ISPs.

Roteiro de Atividades 9

Atividade 9.1 – Instalação do OpenVPN

Pré-condição

1. As interfaces de rede nas máquinas virtuais devem estar configuradas como placa em modo BRIDGE.
2. Nesta atividade, os alunos devem trabalhar em dupla. Uma máquina virtual deverá ser configurada como servidor (Matriz), enquanto a outra deverá ser o cliente (Filial). Para permitir o roteamento entre diferentes redes utilizando o túnel, será necessário criar uma interface de rede virtual adicional em cada máquina virtual. Como regra, a máquina servidora OpenVPN deverá ser configurada com uma interface de rede adicional, cujo endereço será 11.X.0.1 / 255.255.0.0, e a máquina cliente com o endereço 11.Y.0.1 / 255.255.0.0.

> Pergunte ao instrutor os valores de X e Y. Observe que esses valores devem ser exclusivos para cada dupla em laboratório.

3. Para configurar uma interface de rede virtual, edite o arquivo */etc/network/interfaces* e adicione ao final:

```
auto eth0:0

iface eth0:0 inet static

address 11.X.0.1

netmask 255.255.0.0

broadcast 11.X.255.255

network 11.X.0.0
```

4. Reinicie a interface de rede:

```
#/etc/init.d/networking restart
```

Configuração

1. Para instalar o OpenVPN em uma máquina Debian, basta executar:

```
# apt-get install openvpn
```

2. Após instalar o pacote, verifique se o módulo *tun* foi carregado pelo kernel do sistema:

```
# modprobe tun

# lsmod | grep tun
```

3. Vamos gerar a chave de encriptação para uso nos túneis VPN. Lembre-se de que essa chave deve ser transferida para a máquina cliente. Para isso execute:

```
#cd /etc/openvpn

#openvpn --genkey --secret vpnsenha
```

4. Vamos configurar o servidor. Para isso crie o arquivo */etc/openvpn/matriz.conf* com o seguinte conteúdo (obs.: o arquivo segue com comentários):

```
# configura o modulo para a interface
dev tun0
# primeiro ip local tunelado e segundo ip tunelado
ifconfig 172.16.0.1 172.16.0.2
# configura para o modo servidor
proto tcp-server
# define a chave de criptografia
secret /etc/openvpn/vpnsenha
# define a porta a ser utilizada por esse tunel.
# Diferentes tuneis exigem diferentes portas
port 5003
# Testar a conexao
ping 3
ping-restart 120
persist-tun
persist-key
link-mtu 1500
float
# Nivel de log
# 0 - modo quieto exceto para errros fatais
# 1 - parcialmente quieto, mas ira mostrar erros nao fatais
# 3 - saida media, bom para operacao normal
# 9 - verbose ativo, bom para descobrir erros
verb 3
# Ativar compactacao
comp-lzo
# Definir a rota para uma determinada interface de rede
route 11.2.0.0 255.255.0.0 172.16.0.2
```

5. Agora vamos configurar a máquina cliente; para isso será necessário transferir a chave da máquina servidora. Execute:

```
# scp <ip_servidor>:/etc/openvpn/vpnsenha /etc/openvpn/vpnsenha
```

6. Vamos configurar o cliente, para isso crie o arquivo */etc/openvpn/cliente.conf* com o seguinte conteúdo (o arquivo segue com comentários):

```
# configura o modulo para a interface
dev tun0
# chamando o servidor
remote <IP_DO_SERVIDOR>
# primeiro ip local tunelado e segundo ip tunelado
ifconfig 172.16.0.2 172.16.0.1
# configura para o modo servidor
proto tcp-client
# define a chave de criptografia
secret /etc/openvpn/vpnsenha
# define a porta a ser utilizada por esse tunel.
# Diferentes tuneis exigem diferentes portas
port 5003
# Testar a conexao
ping 3
ping-restart 120
persist-tun
persist-key
link-mtu 1453
float
# Nivel de log
# 0 - modo quieto exceto para errros fatais
# 1 - parcialmente quieto, mas ira mostrar erros nao fatais
# 3 - saida media, bom para operacao normal
# 9 - verbose ativo, bom para descobrir erros
verb 3
# Ativar compactacao
comp-lzo
# Definir a rota para uma determinada interface de rede
route 11.1.0.0 255.255.0.0 172.16.0.1
```

7. Para finalizar, reinicie o OpenVPN na máquina servidor e na máquina cliente:

```
#/etc/init.d/openvpn restart
```

Testando a configuração

Para testar o funcionamento, basta executar um *ping* na interface virtual da outra máquina. Por exemplo, a partir do console do servidor, execute:

```
# ping <Endereço ETH0:0 da máquina cliente/servidora>
```

Atividade 9.2 – Utilizando certificados digitais

O OpenVPN implementa o recurso de criptografia utilizando certificados digitais SSL/TLS. Vejamos a seguir a implementação dessa funcionalidade.

Instalação e configuração da Autoridade Certificadora (CA)

1. Primeiro vamos instalar o OpenSSL:

```
#apt-get install openssl
```

2. Crie em */etc/openvpn/* os diretórios:

```
# mkdir /etc/openvpn/demoCA

# mkdir /etc/openvpn/demoCA/certs

# mkdir /etc/openvpn/demoCA/newcerts

# mkdir /etc/openvpn/demoCA/crl

# mkdir /etc/openvpn/demoCA/private
```

3. Dentro do diretório “demoCA”, crie os seguintes arquivos:

```
# pwd

/etc/openvpn/demoCA

# touch index.txt

#echo 01 > serial
```

4. Ajuste a permissão do diretório da Autoridade Certificadora (CA):

```
# chmod -R 700 /etc/openvpn/demoCA
```

5. Gere a chave privada da CA com validade de 10 anos; execute os comandos dentro do diretório da demoCA: */etc/openvpn/demoCA*:

```
# openssl req -nodes -new -x509 -keyout cakey.pem -out cacert.pem
-days 3650

Generating a 1024 bit RSA private key

..........++++++

......++++++

writing new private key to ‘cakey.pem’

-----
```

```
You are about to be asked to enter information that will be
incorporated

into your certificate request.

What you are about to enter is what is called a Distinguished Name
or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [AU]:BR

State or Province Name (full name) [Some-State]:Rio de Janeiro

Locality Name (eg, city) []:Rio de Janeiro

Organization Name (eg, company) [Internet Widgits Pty Ltd]:RNP

Organizational Unit Name (eg, section) []:ESR

Common Name (eg, YOUR name) []:ca.rnp.br

Email Address []:admin@rnp.br
```

6. Verifique se a chave *ca.key* foi gerada com sucesso:

```
# cat /etc/openvpn/demoCA/cakey.pem

Mova a chave [cakey.pem] para o diretório private:

# pwd

/etc/openvpn/demoCA/

# mv cakey.pem ./private
```

Gerar Certificado/Chave da Matriz

```
# cd /etc/openvpn

# openssl req -nodes -new -keyout matriz.key -out matriz.csr

Generating a 1024 bit RSA private key

..........++++++

......++++++

writing new private key to 'matriz.key'

-----

You are about to be asked to enter information that will be
incorporated

into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:Rio de Janeiro
Locality Name (eg, city) []:Rio de Janeiro
Organization Name (eg, company) [Internet Widgits Pty Ltd]:RNP
Organizational Unit Name (eg, section) []:ESR
Common Name (eg, YOUR name) []:matriz.rnp.br
Email Address []:admin@rnp.br
```

Assinar o certificado da Matriz

```
# openssl ca -out matriz.crt -in matriz.csr
Using configuration from /usr/lib/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: May 13 14:41:05 2008 GMT
            Not After : May 13 14:41:05 2009 GMT
        Subject:
            countryName               = BR
            stateOrProvinceName       = Rio de Janeiro
            organizationName          = RNP
            organizationalUnitName    = ESR
            commonName                = matriz.rnp.br
            emailAddress              = admin@rnp.br
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
```

```
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                81:A7:06:D3:C0:3B:7C:4B:17:71:FD:A6:CA:07:AA:CB:44:16:
7F:2A
            X509v3 Authority Key Identifier:
                keyid:1B:F7:17:4E:0E:0D:EE:68:AA:8F:15:62:23:30:61:D8:
B3:0C:2A:B1


Certificate is to be certified until May 13 14:41:05 2009 GMT (365
days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Gerar Certificado/Chave do Cliente

```
#cd /etc/openvpn
# openssl req -nodes -new -keyout cliente.key -out cliente.csr
Generating a 1024 bit RSA private key
..++++++
.++++++
writing new private key to 'cliente.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:Rio de Janeiro
```

```
Locality Name (eg, city) []:Rio de Janeiro
Organization Name (eg, company) [Internet Widgits Pty Ltd]:RNP
Organizational Unit Name (eg, section) []:ESR
Common Name (eg, YOUR name) []:cliente.rnp.br
Email Address []:admin@rnp.br

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Assinar chave do Cliente

```
ads003:/etc/openvpn# openssl ca -out cliente.crt -in cliente.csr
Using configuration from /usr/lib/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 2 (0x2)
        Validity
            Not Before: May 13 14:48:03 2008 GMT
            Not After : May 13 14:48:03 2009 GMT
        Subject:
            countryName               = BR
            stateOrProvinceName       = Rio de Janeiro
            organizationName          = RNP
            organizationalUnitName    = ESR
            commonName                = cliente.rnp.br
            emailAddress              = admin@rnp.br
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
```

```
                28:D4:60:A8:5C:96:86:1E:24:CF:3E:F3:41:A5:92:33:1C:CA
:B1:61
            X509v3 Authority Key Identifier:
                keyid:1B:F7:17:4E:0E:0D:EE:68:AA:8F:15:62:23:30:61:D8:
B3:0C:2A:B1


Certificate is to be certified until May 13 14:48:03 2009 GMT (365
days)
Sign the certificate? [y/n]:y



1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Gerar o protocolo DH – Diffie Hellman

Algoritmo usado para a troca de chaves criptográficas durante a execução do OpenVPN. Utilize no comando a criptografia que consta no arquivo *openssl.cnf*, no caso de 1024 bits:

```
# openssl dhparam -out dh1024.pem 1024
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.................................
```

Configuração do OpenVPN – Matriz

Para cada túnel deverá ser criado um arquivo de configuração. Este arquivo *openvpn* vai ler/executar quando for iniciado, configurando os IPs que serão assumidos na Matriz e no Cliente.

```
# mcedit /etc/openvpn/matriz.conf
# configura o modulo para a interface
dev tun0
# primeiro ip local tunelado e segundo ip tunelado
ifconfig 172.16.0.1 172.16.0.2
#Ativa e configura o protocolo TLS
--tls-server
dh /etc/openvpn/dh1024.pem # protocolo DH
ca /etc/openvpn/demoCA/cacert.pem # Certificado CA
cert /etc/openvpn/matriz.crt # Certificado Matriz
```

```
key /etc/openvpn/matriz.key # chave certificado Matriz


# define a porta a ser utilizada por esse tunel.
# Diferentes tuneis exigem diferentes portas
port 1194
--proto tcp-server
# Testar a conexao
ping 3
ping-restart 120
persist-tun
persist-key
--tun-mtu 1500
float
# Nivel de log
# 0 - modo quieto exceto para errros fatais
# 1 - parcialmente quieto, mas ira mostrar erros nao fatais
# 3 - saida media, bom para operacao normal
# 9 - verbose ativo, bom para descobrir erros
verb 3
# Ativar compactacao
comp-lzo
# Definir a rota para uma determinada interface de rede
route 11.2.0.0 255.255.0.0 172.16.0.2
```

Configuração do OpenVPN – Filial

1. Copie através de um canal seguro o certificado da CA, chave e certificado do cliente e o protocolo DH gerados no servidor (arquivos: *cacert.pem*, *cliente.key*, *cliente.crt* e *dh1024.pem*).

> Use o comando *scp* (mais detalhes com *man scp*) para copiar os arquivos mencionados do servidor da Matriz para o servidor Cliente.

2. Ajuste as permissões da chave (filial.key):

```
# chmod 600 cliente.key
```

3. O arquivo de configuração do cliente é bem parecido com o arquivo de configuração da Matriz; no entanto, preste atenção nos detalhes:

```
#mcedit /etc/openvpn/filial.conf


# configura o modulo para a interface
dev tun0
# chamando o servidor
remote <IP_DO_SERVIDOR>
# primeiro ip local tunelado e segundo ip tunelado
ifconfig 172.16.0.2 172.16.0.1
#Ativa e configura o protocolo TLS
--tls-client
dh /etc/openvpn/dh1024.pem # protocolo DH
ca /etc/openvpn/cacert.pem # Certificado CA
cert /etc/openvpn/cliente.crt # Certificado Matriz
key /etc/openvpn/cliente.key # chave certificado Matriz

# define a porta a ser utilizada por esse tunel.
# Diferentes tuneis exigem diferentes portas
port 1194
--proto tcp-client
# Testar a conexao
ping 3
ping-restart 120
persist-tun
persist-key
--tun-mtu 1500
float
# Nivel de log
# 0 - modo quieto exceto para errros fatais
# 1 - parcialmente quieto, mas ira mostrar erros nao fatais
# 3 - saida media, bom para operacao normal
# 9 - verbose ativo, bom para descobrir erros
```

```
verb 3

# Ativar compactacao

comp-lzo

# Definir a rota para uma determinada interface de rede

route 11.1.0.0 255.255.0.0 172.16.0.1# configura o modulo para a
interface
```

4. Para finalizar, reinicie o OpenVPN na máquina servidor e na máquina cliente:

```
#/etc/init.d/openvpn restart
```

Testando a configuração

Para testar o funcionamento, basta executar um *ping* na interface virtual da outra máquina. Por exemplo, a partir do console do servidor, execute:

```
# ping <Endereço ETH0:0 da máquinacliente/servidora>
```

10

IPSec

objetivos

Introduzir a necessidade de utilização do IPSec em redes IP; apresentar o funcionamento, explicando o modo de trabalho dos protocolos AH, ESP e IKE; definir o uso do IPSec para criação de VPNs em modo túnel e modo transporte; introduzir os problemas relacionados ao uso de NAT em redes VPN IPSec e discutir as soluções utilizadas.

conceitos

IPSec e VPN; instalação e configuração do servidor Shorewall.

Introdução ao IPSec

- Desenvolvido pelo IETF, agrega conceitos de autenticação e privacidade, de maneira transparente, a todas as aplicações que fazem uso da pilha TCP/IP.
- Elevado nível de independência e modularidade entre os protocolos.
- Os principais protocolos responsáveis por garantir níveis de segurança são:
  - Authentication Header (AH).
  - Encapsulating Security Payload (ESP).

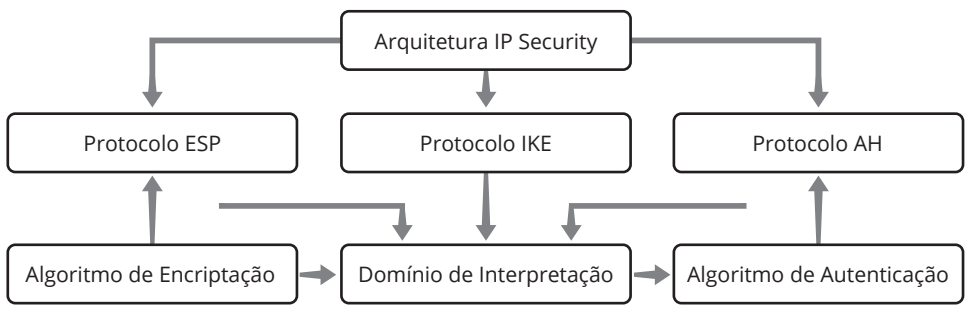


Figura 10.1
A Arquitetura IP Security.

O Internet Protocol Security (IPSec) é uma especificação desenvolvida pela Internet Engineering Task Force (IETF) com o objetivo de prover uma arquitetura segura na camada de rede dos protocolos IPv4 e IPv6. Sua principal vantagem refere-se ao fato de prover (de maneira transparente) autenticação, integridade, controle de acesso e confidencialidade às aplicações já desenvolvidas pela organização.

Outra característica importante é o elevado nível de independência e modularidade entre os protocolos. Os algoritmos de criptografia utilizados nos protocolos (RSA, DES etc.) e os mecanismos de distribuição de chaves (IKE) são independentes entre si. Tal característica permite o aperfeiçoamento e a substituição de novos protocolos e algoritmos sem, no entanto, comprometer a arquitetura já definida. A negociação pelo transmissor e receptor dos protocolos que serão utilizados na comunicação é conhecida como Associação de Segurança – AS (Security Association – SA) e é identificada de forma única por três parâmetros: o Security Parameter Index (SPI), o endereço IP de destino e o identificador do protocolo (AH ou ESP).

Os dois protocolos responsáveis por garantir níveis de segurança no IPSec são:

- **AH (Authentication Header)**: oferece integridade e autenticação, sem se preocupar com questões de privacidade;
- **ESP (Encapsulating Security Payload)**: oferece primordialmente privacidade. Naturalmente os protocolos AH e ESP podem ser utilizados em conjunto, se necessário, ou separadamente.

Para os dois cabeçalhos, o IETF não determina os algoritmos de segurança que devem ser utilizados. De forma geral, são recomendados os algoritmos MD5 e SHA para integridade e autenticação, e o DES para encriptação.

Associações de segurança

Uma associação de segurança é definida pelo algoritmo, modo de operação, chave e demais propriedades de segurança.

Uma vez que existe um elevado nível de independência e modularidade entre os protocolos (AH e ESP), alguma forma de associação deve existir entre eles para permitir que o emissor e o receptor saibam quais serão os mecanismos e chaves utilizados para a comunicação. Para este fim é criada uma associação de segurança que define, basicamente, o algoritmo, o modo de operação, a chave e as demais propriedades aplicadas sobre os pacotes processados.

Uma associação de segurança é identificada de forma única e é composta, basicamente, por três parâmetros: o Security Parameter Index (SPI), o endereço IP de destino e o identificador do protocolo (AH ou ESP):

- **SPI**: trata-se de um número único que identifica a AS gerada durante a fase de negociação que antecede o estabelecimento da conexão. Todos os membros envolvidos no processo de comunicação devem conhecer esse número e utilizá-lo sempre que transmitirem uma informação.
- **Endereço IP do destino**: pode ser unicast, broadcast ou ainda multicast. No entanto, para a definição dos mecanismos de gerenciamento de AS, o IPSec assume um endereço de destino unicast, estendendo as definições para os casos de broadcast e multicast.
- **Identificador do protocolo**: trata-se de um número que identifica se o protocolo utilizado é o AH (número 51) ou o ESP (número 50).

Observe que a negociação para criação de um AS entre emissor e receptor envolve, também, a definição das chaves, dos algoritmos e dos padrões utilizados por estes algoritmos. As associações são unidirecionais; dessa forma, sessões de comunicação entre duas máquinas terão normalmente um SPI para cada direção de tráfego.

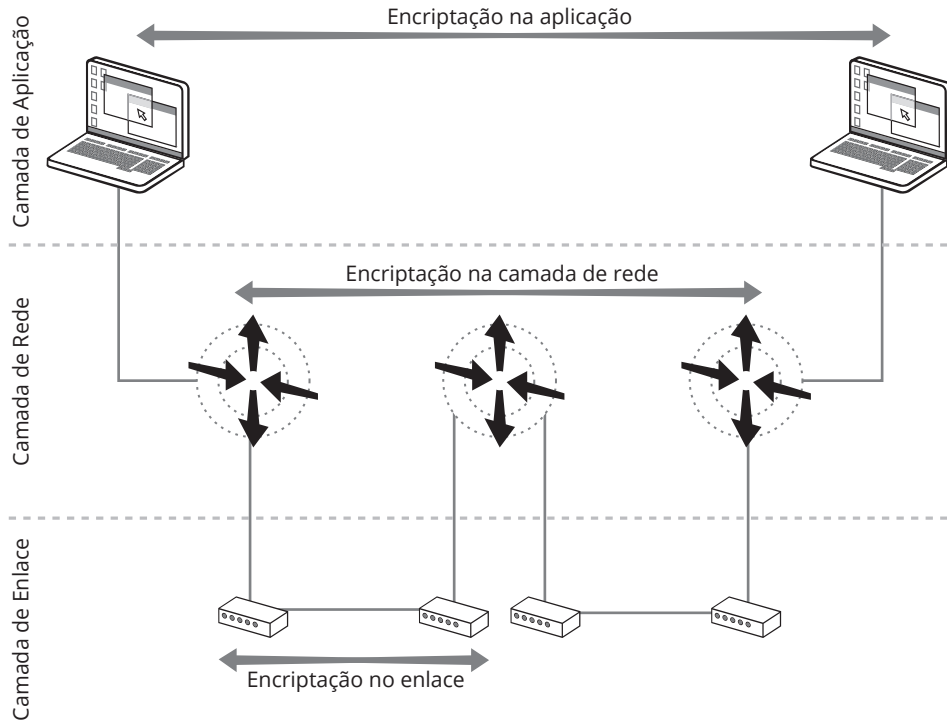


Figura 10.2
Associações de segurança.

Modos de operação

- Modo Transporte (host – host).
- Modo Túnel (host – gateway – gateway – host).

Uma AS pode ser estabelecida de dois modos diferentes: transporte ou túnel.

- **Modo Transporte:** a Associação de Segurança é estabelecida entre dois hosts preservando, dessa forma, o cabeçalho IP original. No caso do ESP, uma AS em modo transporte provê serviços de segurança somente para os protocolos de nível mais alto, não incluindo o cabeçalho IP ou os cabeçalhos de extensão que precedem o ESP. No entanto, o AH estende a proteção a esses cabeçalhos. Isso se deve ao fato de o ESP cifrar os dados que o sucedem no pacote, além de autenticar apenas a "porção ESP" do pacote, enquanto que o AH autentica o pacote todo.

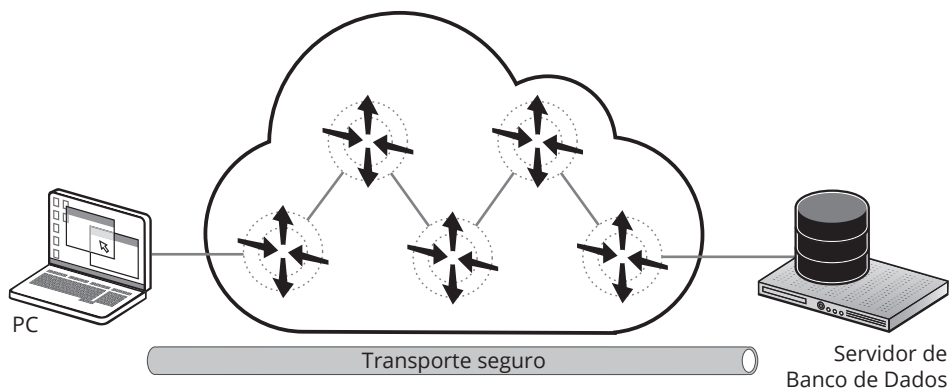
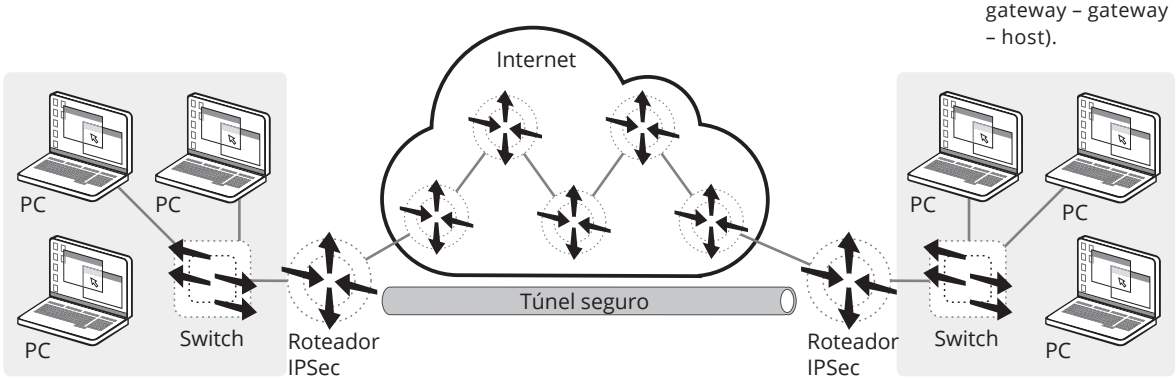


Figura 10.3
Modo transporte (host – host).

- **Modo Túnel:** trata-se de uma associação aplicada a um túnel IP. Neste modo os pacotes de uma rede são totalmente encapsulados pelo gateway IPSec, ou seja: um pacote TCP/IP de um host da rede interna, com destino a um host de uma rede remota, recebe um novo cabeçalho TCP/IP como endereço de origem do gateway interno, e como destino o gateway IPSec do receptor.

Figura 10.4
Modo Túnel (host – gateway – gateway – host).



Protocolo AH (IP Authentication Header)

Objetivo: garantir a autenticidade do pacote e que ele não foi alterado durante a transmissão

Previne ataques como:

- Replay: quando o atacante intercepta um pacote válido e autenticado pertencente a uma conexão, replica-o e reenvia-o.
- Spoofing: quando o atacante assume o papel de uma máquina confiável para o destino e, dessa forma, ganha privilégios na comunicação.
- "Roubo de conexões" (connection hijacking): quando o atacante intercepta um pacote no contexto de uma conexão e passa a participar da comunicação.

O objetivo do AH é oferecer autenticação e integridade aos datagramas IP, ou seja, garantir a autenticidade do pacote e que ele não foi alterado durante a transmissão. Independentemente do algoritmo utilizado, o hash da mensagem é calculado sobre todo o datagrama IP. Isso inclui não apenas o cabeçalho IP, mas também todos os demais cabeçalhos de outros protocolos e os dados transportados.

Para evitar possíveis erros, os campos e opções do cabeçalho IP, que podem ser modificados ao longo do caminho entre sua origem e destino, são preenchidos com zeros pelo algoritmo para efeito de cálculo.

O uso do AH previne ataques do tipo:

- **Replay**: quando o atacante intercepta um pacote válido e autenticado pertencente a uma conexão, replica-o e o reenvia, "entrando na conversa". A utilização do campo "Sequence Number" ajuda na prevenção a esse tipo de ataque, pois permite numerar os pacotes que trafegam dentro de uma determinada associação de segurança.
- **Spoofing**: ocorre quando o atacante assume o papel de uma máquina confiável para o destino e, dessa forma, ganha privilégios na comunicação. A utilização de mecanismos de autenticação previne este tipo de ataque.

- **"Roubo de conexões" (connection hijacking)**: ocorre quando o atacante intercepta um pacote no contexto de uma conexão e passa a participar da comunicação. A utilização de mecanismos de autenticação previne este tipo de ataque.

Formato do AH (Authentication Header)

AH deve estar localizado logo após os cabeçalhos que são examinados por cada roteador intermediário entre uma origem e um destino qualquer, ou seja, após o próprio cabeçalho do protocolo IPv4.

A figura a seguir ilustra o cabeçalho de autenticação, cujos campos transportam informação necessária à autenticação de datagramas IP. Os seus campos são:

- **Próximo cabeçalho (Next Header)**: identifica o cabeçalho agregado a este;
- **Tamanho do módulo (Payload Length)**: contém o tamanho do módulo dos dados;
- **Índice de parâmetros de segurança (Security Parameter Index)**: número arbitrário de 32 bits que indica ao receptor o conjunto de informação necessária para estabelecer um canal de comunicação seguro;
- **Número de sequência (Sequence Number)**: contador incrementado cada vez que um pacote é enviado para o mesmo endereço, com o mesmo SPI. Permite identificar o pacote e o número de pacotes enviados com os mesmos parâmetros de segurança. Permite assim evitar ataques replay, ou seja, cópias de pacotes e envio fora de sequência que pode confundir a comunicação;
- **Dados de autenticação (Authentication Data)**: campo que contém os dados de autenticação.

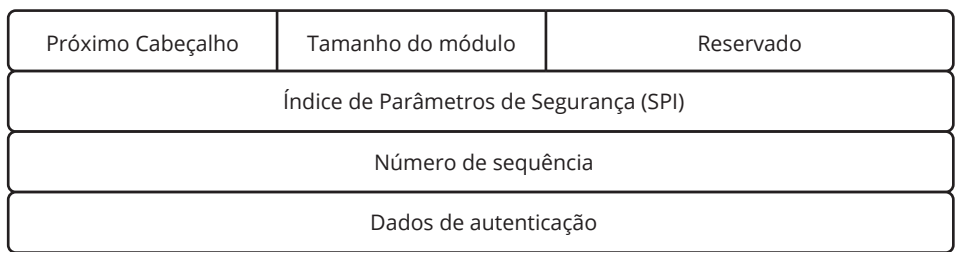


Figura 10.5
Authentication Header.

Modos de funcionamento do AH

Opera nos dois modos de funcionamento IPSec:

- Modo Transporte.
- Modo Túnel.

O AH pode ser utilizado nos dois modos de funcionamento do IPSec: modo Transporte ou Túnel, como exemplificado nas figuras a seguir. O modo de transporte é usado quando se pretende ter um canal seguro entre dois sistemas terminais. Apresenta menos overhead de processamento mas não autentica os campos alteráveis do datagrama. O modo de túnel é usado quando se pretende estabelecer comunicação segura com um sistema intermediário. Esse modo apresenta maior overhead, mas permite autenticar todo o datagrama.

O cabeçalho de autenticação não providencia confidencialidade ou proteção contra ataques de análise de tráfego, e por isso é utilizado apenas com o cabeçalho de encapsulamento de dados de segurança.

Para manter a interoperabilidade entre produtos IPSec de diferentes fabricantes, são recomendados como algoritmos de cabeçalho o HMAC-MD5 e HMAC-SHA1.

Figura 10.7
Modo túnel.

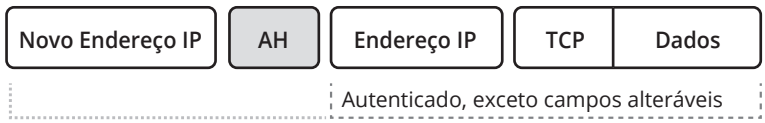


Figura 10.6
Modo de transporte.

Protocolo ESP (IP Encapsulating Security Payload)

O objetivo primordial do ESP é oferecer privacidade ao conteúdo dos dados encapsulados em um datagrama IP. Dependendo do algoritmo e do seu modo de operação (transformação), também podem ser oferecidas autenticação e integridade. No entanto, será mostrado adiante que tais transformações não são necessariamente seguras. Portanto, sempre que for requerida autenticação e/ou integridade, o AH deve ser utilizado em conjunto com o ESP.

Formato do ESP

O ESP deve estar localizado logo após os demais cabeçalhos IP, incluindo o próprio AH. O cabeçalho de Encapsulamento de Dados de Segurança implementa integridade e confidencialidade aos datagramas IP, através da cifra dos dados contidos no datagrama. Pode ainda garantir autenticação, dependendo do algoritmo e do modo utilizados. No entanto, não garante a irretratabilidade.

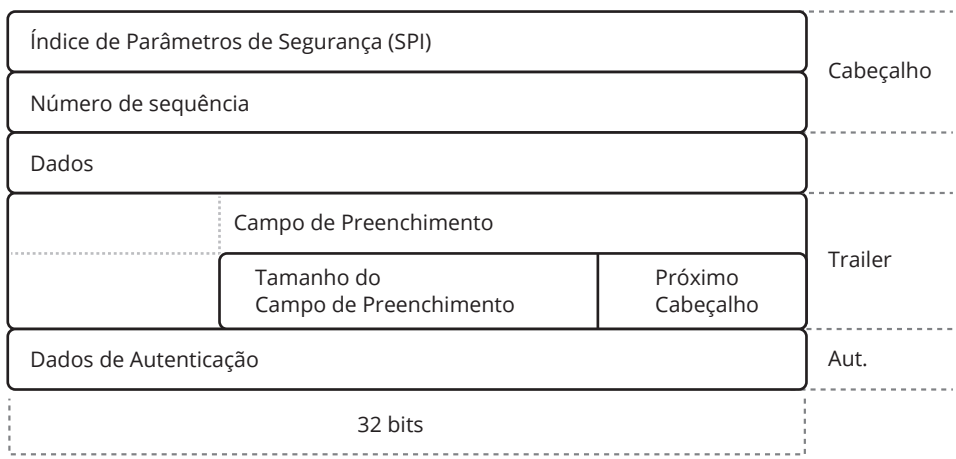


Figura 10.8
Formato do ESP.

O ESP é constituído por seis campos, conforme a figura anterior:

- **Índice de parâmetros de segurança (Security Parameter Index)**: número arbitrário de 32 bits que diz ao receptor o grupo de protocolos de segurança que o emissor está utilizando para estabelecer uma comunicação segura;
- **Número de sequência (Sequence Number)**: contador usado para proteção de replay;
- **Dados (Payload Data)**: contém os dados transportados pelo datagrama;

- **Campo de preenchimento (Padding)**: representa o preenchimento necessário a determinados tipos de algoritmos de cifra, que podem necessitar que o tamanho de dados seja múltiplo de um certo número de bytes;
- **Tamanho do campo de preenchimento (Pad Length)**: indica a quantidade de dados que é simplesmente de preenchimento;
- **Próximo cabeçalho (Next Header)**: identifica o tipo do próximo cabeçalho;
- **Dados de autenticação (Authentication Data)**: esse campo contém a assinatura digital calculada sobre os dados restantes do cabeçalho. O seu tamanho varia, dependendo do tipo de algoritmo de autenticação utilizado.

Modos de funcionamento do ESP

Pode ser também utilizado nos dois modos de funcionamento da IPSec:

- Modo Transporte.
- Modo Túnel.

O cabeçalho de encapsulamento de dados de segurança pode ser também utilizado nos dois modos de funcionamento da IPSec, como representado nas figuras a seguir, e ainda em conjunto com o cabeçalho de autenticação de dados. À medida que a cifra envolve cálculos computacionais mais complexos, a utilização desse cabeçalho introduz maior overhead de processamento nos sistemas terminais do canal seguro. No entanto, não afeta possíveis sistemas intermediários. O custo da utilização desse cabeçalho varia com as especificidades das implementações, como algoritmo de cifra utilizado ou tamanho da chave. As implementações em hardware do algoritmo de cifra serão úteis em casos em que se pretenda obter melhor desempenho.

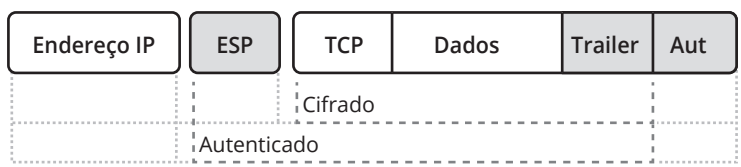


Figura 10.9
O modo Transporte.

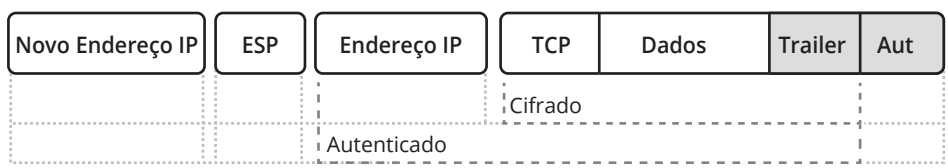


Figura 10.10
O modo Túnel.

O ESP foi desenhado para utilizar qualquer tipo de protocolo de criptografia. No entanto, o IPSec define padrões de uso para os protocolos: DES-Cipher Block Chaining Mode para a cifra e HMAC-SHA1 e o HMAC-MD5 para a autenticação, de modo a garantir interoperabilidade mínima entre diferentes implementações.

Gerenciamento de chaves

Fornece o mecanismo necessário para negociar os algoritmos de criptografia e as chaves que serão utilizadas na troca de dados. Pode ser realizado das seguintes formas:

- Gestão manual.
- Gestão automática.

Os protocolos AH e ESP são a base do IPSec e permitem proteger os dados. No entanto, para que exista comunicação segura, é necessário existir um terceiro mecanismo que permita negociar os algoritmos e as chaves de criptografia que serão utilizados por esses protocolos para transferir os dados. O grupo de trabalho do IETF responsável por definir o padrão determinou que os sistemas compatíveis devem suportar tanto uma Associação de Segurança manual como uma automatizada. Dessa forma, a gestão de chaves pode ser implementada:

- **Gestão manual**: o administrador configura manualmente cada sistema, fornecendo as chaves e as informações para o gerenciamento da AS relevantes em uma comunicação segura com os outros sistemas. As técnicas manuais têm seu uso recomendado para ambientes estáticos pequenos, mas esse método não é muito prático em redes maiores;
- **Gestão automática**: utiliza protocolos automatizados de gerenciamento de chave. O gerenciamento automatizado também oferece bastante escalabilidade para sistemas distribuídos maiores que ainda estejam em desenvolvimento. O usuário pode ainda utilizar vários protocolos em um gerenciamento automatizado.

Qualquer produto IPSec apresenta a possibilidade de gestão manual de chaves. No entanto, esse é um processo moroso e pouco viável, pois as chaves poderão ser facilmente obtidas.

Gerência automática de chaves

Internet Key Exchange (IKE) é composto pelos seguintes algoritmos:

- ISAKMP: providencia um framework para autenticação e troca de chaves.
- Oakley: descreve uma série de procedimentos para troca de chaves, conhecida como "modo".
- SKEME: descreve um mecanismo versátil para troca de chaves proporcionando anonimato, não repúdio e a atualização rápida.

O conjunto de protocolos Internet Key Exchange (IKE) é utilizado pelo IPSec para definir e realizar a troca de chaves de forma automática e segura, sendo implementado sobre UDP porta 500. É composto pelos seguintes algoritmos:

- **ISAKMP**: providencia um framework para autenticação e troca de chaves, porém não define seu funcionamento. O ISAKMP é desenhado para possuir um mecanismo de troca independente construído para fornecer suporte a diferentes tipos de algoritmos.
- **Oakley**: descreve uma série de procedimentos para troca de chaves, conhecida como "modo", e detalha os serviços fornecidos por eles (exemplo: perfect forward secrecy para chaves, proteção da identidade e autenticação).
- **SKEME**: descreve um mecanismo versátil para troca de chaves, proporcionando anonimato, não repúdio e atualização rápida.

Sua principal aplicação é estabelecer associações de segurança (SAs) IPSec para os protocolos AH e ESP.

O modelo de implantação geral do IKE é mostrado na figura a seguir. Os mecanismos IPSec e IKE são separados em módulos. Quando não existe uma associação de segurança para um pacote que precisa ser processado (enviado ou recebido), o mecanismo IPSec entra em contato com o mecanismo do IKE e solicita que seja estabelecida uma SA adequada. Uma vez terminado o handshake do IKE, é registrada a SA no mecanismo IPSec.

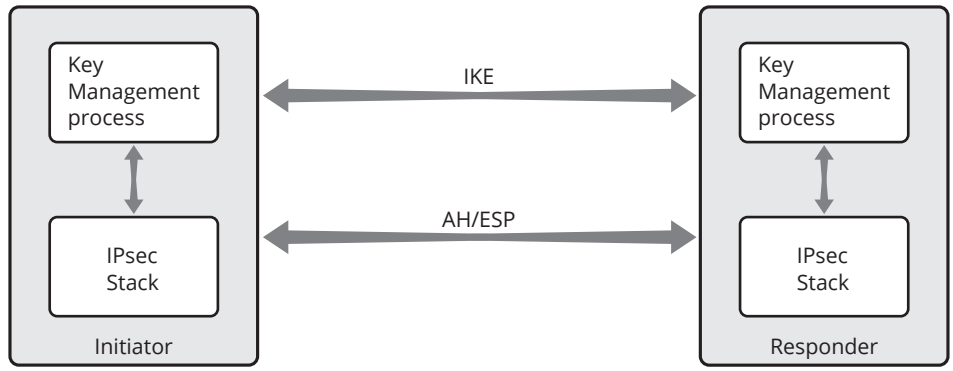


Figura 10.11
Ambiente de operação.

Além disso, o tráfego IKE entre os pares (hosts) pode ser utilizado para atualizar o material da chave ou ajustar parâmetros de operação, como algoritmos.

Principais conceitos sobre o IKE

- Inicializador (Initiator) e Respondedor (Responder).
- Perfect Forward Secrecy.
- Resistência ao Denial of Service.
- Uso de chaves para autenticação.
- Proteção da identidade.

Inicializador (initiator) e Respondedor (Responder)

Embora o IPsec seja basicamente simétrico, o IKE não é. A parte que envia a primeira mensagem é chamada de INITIATOR. A outra parte é chamada de RESPONDER. No caso de conexões TCP, o INITIATOR será tipicamente o par que iniciou a conexão (ou seja, o cliente).

Perfect Forward Secrecy

Uma das principais preocupações na implementação do IKE é que o tráfego seja protegido, mesmo se o material que compõe a chave de um dos nós seja comprometido após o término da conexão. Essa propriedade é frequentemente conhecida como Perfect Forward Secrecy (PFS).

Resistência ao Denial of Service

Como o IKE permite aos pares (hosts) inicializar operações criptográficas computacionalmente caras, existe a possibilidade de que os recursos do par (host) sejam consumidos totalmente, provocando um ataque de negação de serviço (DoS). O IKE inclui contramedidas criadas para minimizar este risco.

Chaves para autenticação

Como as Associações de Segurança são essencialmente simétricas, ambas as partes devem, em geral, ser autenticadas. O IKE precisa ser capaz de estabelecer SAs entre uma grande quantidade de pares (hosts) com vários tipos de relacionamento e prioridades. Para isso é implementado um modelo flexível onde os hosts podem se autenticar via compartilhamento da chave e/ou assinaturas digitais.

Proteção da identidade

Embora o IKE obrigue a autenticação mútua entre os pares envolvidos na comunicação, foi considerada importante, pelo grupo de trabalho que especificou o protocolo, que a identidade dos hosts envolvidos fosse protegida em alguns casos. Em particular, os pares devem ser capazes de ocultar a sua identidade para algum observador externo passivo, e um dos pares deve estar em condições de exigir que o autor, para autenticar, autentique a si mesmo. Nesse caso, os desenvolvedores escolheram fazer com que a parte que fala primeiro (INITIATOR) seja a primeira a se identificar.

Antes de detalharmos o modo de trabalho do protocolo IKE, vamos entender o funcionamento do algoritmo Diffie-Hellman (DH), que é a base para a proteção da identidade.

Algoritmo Diffie-Hellman (DH)

O IPSec utiliza a negociação Diffie-Hellman para criar uma chave de sessão (simétrica) entre os hosts da comunicação segura. O protocolo Diffie-Hellman é composto por três fases:

- **Fase 1**: cada host gera uma chave pública a partir de parâmetros pré-combinados (Diffie-Hellman parameters) e um número aleatório secreto.
- **Fase 2**: os hosts trocam as chaves públicas.
- **Fase 3**: a chave de sessão é calculada a partir das chaves públicas e dos números aleatórios secretos.

O processo matemático utilizado pode ser resumido em 4 passos (veja a figura):

- Cada host obtém os parâmetros Diffie-Hellman (podem ser hard-coded). Um número primo “p” (> 2) e uma base “g” (número inteiro < p)
- Cada host gera um número privado X < (p – 1)
- Cada host gera sua chave pública Y: Y = g^X % p
- Os hosts trocam as chaves públicas e calculam a chave secreta ZZ. Zr = Yi^Xr % p e Zi=Yr ^Xi % p
- Matematicamente ZZ é idêntica para ambos os hosts: Zi = Zr

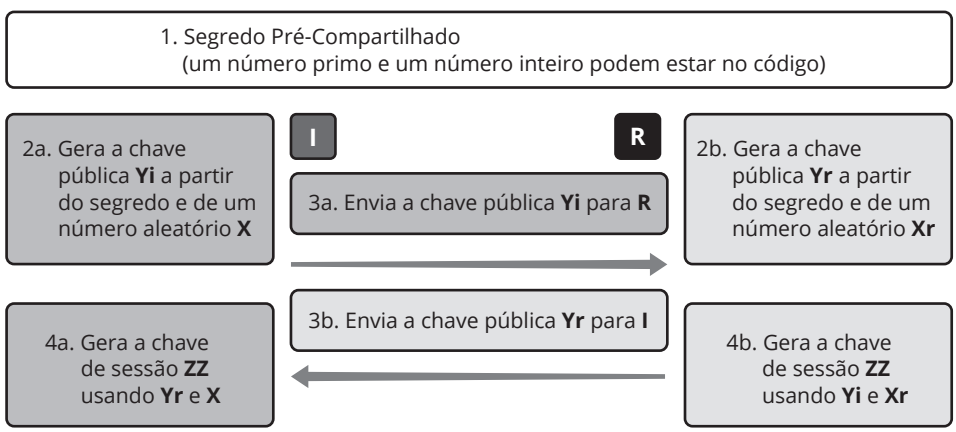

1. Segredo Pré-Compartilhado
(um número primo e um número inteiro podem estar no código)
2a. Gera a chave pública **Yi** a partir do segredo e de um número aleatório **X**
I
R
2b. Gera a chave pública **Yr** a partir do segredo e de um número aleatório **Xr**
3a. Envia a chave pública **Yi** para **R**
3b. Envia a chave pública **Yr** para **I**
4a. Gera a chave de sessão **ZZ** usando **Yr** e **X**
4b. Gera a chave de sessão **ZZ** usando **Yi** e **Xr**

Figura 10.12
Algoritmo Diffie-Hellman (DH).

Visão geral do funcionamento do protocolo

De uma forma geral, existem dois tipos de handshake do IKE:

- Aqueles que estabelecem uma associação de segurança IKE.
- Aqueles que estabelecem uma associação de segurança AH ou ESP.

Quando dois pares que nunca tinham se comunicado antes precisam estabelecer um SA AH / ESP, eles devem primeiro estabelecer uma SA IKE. Isso permitirá que eles troquem uma quantidade de dados de forma protegida. Eles podem usar essa SA para fazer um segundo handshake para estabelecer SAs para o AH e o ESP. Esse processo é mostrado na figura a seguir. A notação E (SA, XXXX) é usada para indicar que o tráfego é criptografado no âmbito de uma dada SA.

Figura 10.13
Visão geral do funcionamento do protocolo (Fase 1 e Fase 2).

Para pensar

A terminologia IKE é um pouco confusa, referindo-se a diferentes circunstâncias de "fases" e "modos". Para entendermos, vamos definir como Fase 1 o estabelecimento da SA do IKE e de Fase 2 a constituição da SAs AH / ESP. Observe que é bem provável que o handshake da Fase 2 seja executado várias vezes, já que a Fase 1 foi finalizada. Isto pode ser útil para o estabelecimento de múltiplas SAs AH / ESP com diferentes propriedades criptográficas.

Fase 1

Main Mode:

- Fornece proteção para a identidade e resistência a ataques de DoS.

Agressive Mode:

- É bem mais rápido que o Main Mode, porém não consegue oferecer os mesmos recurso que este.

Há várias formas de funcionamento do handshake da Fase 1, determinadas de acordo com a utilização pelos diferentes mecanismos de autenticação. Contudo, em termos gerais, o handshake da Fase 1 pode ser classificado em uma destas duas categorias básicas: MAIN MODE, que fornece proteção da identidade e resistência a ataques de DoS, e AGGRESSIVE MODE, que não oferece esses recursos.

Funcionamento do Main Mode

Figura 10.14
Main Mode.

O handshake do Main Mode é composto por seis mensagens (3 round trip). Oferece proteção da identidade e resistência a ataques de DoS. Uma visão geral dos handshakes pode ser observada acima.

No primeiro round trip, o Initiator oferece um conjunto de algoritmos e parâmetros. O Responder escolhe um conjunto que atenda a suas políticas pré-configuradas e responde com esse conjunto. Ela também fornece um CookieR, que será utilizado para prevenir ataques DoS. Nesse ponto, ainda não existe uma associação de segurança, mas os pares (hosts) têm provisoriamente acordados os parâmetros de funcionamento. Esses parâmetros incluem um grupo Diffie-Hellman (DH), que será utilizado no segundo round trip.

No segundo round trip, o Initiator envia informações para a troca da chave. Isso geralmente é constituído pelo Initiator do Diffie-Hellman (Yi). Ele também fornece o CookieR, que foi fornecido pelo Responder. O Responder responde com a sua própria DH (Yr). Nesse ponto, tanto Initiator e Responder podem calcular a chave partilhada DH (ZZ). No entanto, não houve qualquer autenticação e, portanto, eles não sabem com certeza se a conexão não foi comprometida ou atacada. Observe que, enquanto os pares (hosts) geram uma nova chave DH, o algoritmo PFS é acionado para garantir a autenticidade dos pares (hosts).

No round trip final, os pares (hosts) estabelecem suas identidades. Como eles compartilham uma chave (não autenticada), podem enviar suas identidades de forma cifrada, fornecendo assim proteção para a sua identidade. O método exato de provar a identidade depende do tipo de credencial que está sendo utilizado (chave assinada, chave cifrada, segredos compartilhados etc.). Portanto, cada um dos lados poderia fornecer seu certificado e verificar a assinatura associada ao certificado da outra parte comunicante. Exemplo: se for utilizado um algoritmo de chaves compartilhadas, a autenticação consiste apenas em verificar uma ID e uma chave MAC. A autenticação com algoritmos de chave pública segue princípios semelhantes de funcionamento, porém possui algumas características peculiares.

No final do handshake do Main Mode, os pares (hosts) compartilharam:

- Um conjunto de algoritmos de criptografia para uso no IKE;
- Tráfego cifrado e chaves autenticadas;
- Conhecimento mútuo da identidade de cada par (host).

Funcionamento do Aggressive Mode

Embora o Main Mode do IKE forneça os serviços necessários, existe a preocupação de que um grande número de round trips seja exigido para permitir a comunicação, provocando um tempo excessivo de espera. Assim sendo, o Aggressive Mode foi definido utilizando o recurso de enviar mais informações e menos mensagens (mais dados por pacote). Entretanto, essa característica não permite proteger a identidade dos pares (hosts) envolvidos, deixando o sistema vulnerável a ataques de DoS.

Figura 10.15
Funcionamento do Aggressive Mode.

Após o primeiro round trip, os pares (hosts) têm todas as propriedades necessárias para a comunicação, porém o Initiator e o Responder ainda não se autenticaram mutuamente. A terceira mensagem fecha o ciclo de autenticação do Initiator. Observe que a autenticação dos dados é enviada em texto em claro, não sendo possível, por consequência, garantir a proteção da identidade. Como o Responder faz o acordo de chaves DH sem o round trip do Initiator, também não existe proteção contra ataques DOS.

Fase 2

O objetivo do IKE é criar associações para proteger outros tipos de tráfego, não se limitando apenas a estabelecer SAs IKE. A Fase 2 (conhecida como "Quick Mode") é utilizada para este fim. O handshake base da Fase 2 é mostrado na figura a seguir.

Figura 10.16
Fase 2.

Fase 2 com PFS

Na versão anterior do Quick Mode, os pares (hosts) usam o material já definido na Fase 1 para transportar e derivar uma chave temporária, garantindo assim a proteção do tráfego. O "Mode Quick" também permite um novo handshake Diffie-Hellman para o tráfego PFS. Nesse caso, as duas primeiras mensagens mostradas na imagem incluiriam, também, a troca da chave.

Figura 10.17
Fase 2 com PFS.

Network Address Translation (NAT) Transverso

Definição: no protocolo ESP não é possível fazer a tradução de endereço e utilizar a informação de portas de origem e destino como forma de multiplexação das conexões.

NAT ou Network Address Translation é a função de roteamento que permite a tradução de endereços entre redes públicas e privadas. Essa tradução torna-se necessária por causa da limitada oferta de endereços públicos na internet. A principal característica do serviço de NAT é modificar o endereço de origem do pacote IP trocando-o, normalmente, pelo endereço publicamente conhecido utilizado pelo gateway da rede.

Os protocolos de transporte TCP e UDP utilizam o conceito de multiplexação através de portas de origem e destino para viabilizar várias conexões simultâneas. O serviço de NAT deve utilizar um endereço IP público para traduzir vários endereços privados (NAT Masquerade ou NAT Hide), utilizando portas diferentes e armazenando todas estas informações em uma tabela de conexões.

NAT muda o cabeçalho IP exterior.

- IPSec AH não funciona.
- IPSec ESP depende.
- Em modo túnel funciona.
- Em modo transporte não deve funcionar.

Checksums UDP ou TCP.

- Solução: NAT-T (NAT transversal).
- Encapsulamento de datagramas IPSec em datagramas UDP.

Principais problemas apresentados entre os mecanismos NAT e IPSec:

- AH é incompatível com NAT (todo o pacote é autenticado, HMAC);
- NATs não podem atualizar checksums de camadas superiores;
- O número da porta UDP utilizada pelo IKE não pode ser alterado;
- NATs não podem multiplexar streams de dados IPSec;
- A identificação do bloco IKE contém os endereços IP embutidos.

A solução adotada é utilizar o NAT-T (NAT Transversal) que permite encapsular datagramas IPSec em datagramas UDP.

Roteiro de Atividades 10

Atividade 10.1 – Criando o firewall

IPCop é um software livre (código aberto) baseado no GNU/Linux e customizado para funcionar como um firewall e roteador em uma rede de computadores. Trata-se de um poderoso e flexível firewall, que possibilita maior expansão sem acrescentar potenciais vulnerabilidades de segurança para a distribuição base.

Criando a máquina virtual IPCop

Com a ferramenta VirtualBox, crie uma máquina virtual com as seguintes características:

- Nome: IPCop
- Tipo: Linux
- Versão: Other Linux
- Memória: 1420 MB

1. Crie o disco virtual:

- Disco VDI
- HD: 20 GB de espaço

2. Nas configurações, na aba “Redes”, habilite duas placas de rede com a opção “Conectado a Placa em modo Bridge”.
3. Em “Armazenamento”, escolha a imagem .iso do IPCop.

Instalando o sistema IPCop

Pré-instalação

1. Selecione a linguagem: Brazilian Portuguese.
2. Mapeamento do Teclado: br-abnt2.
3. Fuso Horário: Brazil/West.
4. Configure a data.
5. Selecione o disco para instalação.
6. Disk Instalation: OK.
7. Disk Instalation: Hard Disk.
8. Na tela “Restaurar” será perguntado sobre opções de backup; pule essa opção (PULE).
9. Parabéns.
10. Hostname: ipcop
11. Nome do domínio: localhost
12. Interface RED (Interface da rede externa):
    12.1. Marque a opção DHCP.

13. Atribuição da placa:

13.1. Aparecerão duas placas de rede.

13.2. Selecione a primeira, que será ETH0 e RED.

13.3. A segunda será ETH1 e GREEN (Interface da rede interna).

13.4. Depois selecione a opção PRONTO.

14. Interface GREEN:

14.1. Coloque o endereço de sua rede local como indicado no diagrama da Atividade 10.2.

Exemplo: 192.168.X0.1/255.255.255.0

15. Interface RED: PULE.

16. Configurações de DNS e Gateway: 8.8.8.8 e 4.4.4.4.

17. Configuração de servidor de DNS: PULE.

18. Configure senha para root: “rnpesr”

19. Configure senha para admin: “rnpesr”

20. Configure senha para backup: “rnpesr”

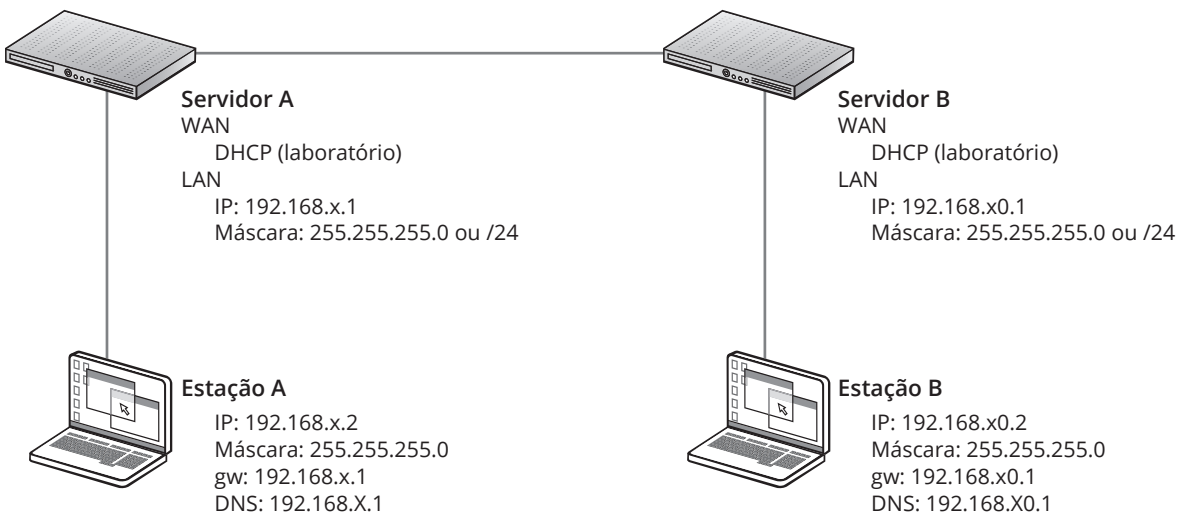
21. OK.

22. Instalação concluída.

Atividade 10.2 – Configurando as interfaces de rede

Pós-instalação

Essa atividade será realizada em dupla. As máquinas devem obedecer à topologia de rede seguinte, onde as máquinas dos alunos estejam em redes diferentes e as máquinas virtuais IPCop sejam seus gateway default. Para simplificar o entendimento, observe o diagrama de rede a seguir:



Pergunte ao instrutor o valor de X, não esquecendo de substituí-lo nos endereços das estações e máquinas virtuais.

Configuração da interface LAN do Linux com o IPCop (Servidor A)

1. Configure a interface da estação de trabalho Windows (Estação A). Nas propriedades de rede da estação de trabalho Windows, selecione a opção “Protocolo TCP/IP” e configure com os parâmetros:

- Endereço IP: 192.168.X.2
- Máscara de subrede: 255.255.255.0
- Gateway padrão: 192.168.X.1

2. Marque a opção “Usar os seguintes endereços de servidor DNS” e informe o servidor de DNS preferencial: 192.168.X.1

3. Clique em “OK”.

Neste momento, a máquina Windows deve ser capaz de realizar um *ping* em qualquer endereço IP externo à rede. Isso acontece porque, por padrão, o IPCop permite o acesso da rede interna à rede externa sem restrições e realiza o NAT na interface WAN.

4. Para testar a conectividade, abra o “Prompt de Comando” (DOS) na máquina Windows e execute:

C:\ping 192.168.X.1
C:\ping www.rnp.br

5. Repita a operação de configuração das interfaces de rede no servidor Linux com IPCop no servidor B.

Lembre-se de que os endereços de rede LAN são diferentes do servidor A (serão fornecidos a resposta e o endereço de acesso).

Configuração da interface da estação de trabalho Windows (Estação B)

1. Nas propriedades de rede da estação de trabalho Windows selecione a opção “Protocolo TCP/IP” e configure com os parâmetros abaixo:

- Endereço IP: 192.168.X0.2
- Máscara de subrede: 255.255.255.0
- Gateway padrão: 192.168.X0.1

2. Marque a opção “Usar os seguintes endereços de servidor DNS” e informe o servidor de DNS preferencial: 192.168.X0.1

3. Clique em “OK”.

Neste momento a máquina Windows deve ser capaz de realizar um *ping* em qualquer endereço IP externo à rede. Isso acontece porque, por padrão, o IPCop permite o acesso da rede interna à rede externa sem restrições e realiza o NAT na interface WAN.

4. Para testar a conectividade, abra o “Prompt de Comando” (DOS) na máquina Windows e execute:

C:\ ping 192.168.X0.1
C:\ping www.rnp.br

Atividade 10.3 – Configurando IPSec no IPCop

IPCop possui várias funcionalidades, entre elas a criação de túneis VPN utilizando o protocolo IPSec. Nessa atividade será configurado o túnel utilizando o IPSec.

Servidores A e B

Abra o navegador web na estação de trabalho Windows e acesse o endereço: https://192.168.X.1:8443/. Serão solicitados usuário e senha (usuário = *root* e senha = *a senha de root do servidor Linux*).

1. No menu, clique em "VPNs" e selecione "IPSec".
2. Marque a opção "IPSec on RED" (a interface externa da rede) e clique em "Salvar".
3. Clique na opção "Adicionar" e depois na opção "Rede Privativa Virtual Rede-à-Rede" e "Selecionar".
4. No servidor A digite o nome "Server A" e no B "Server B".
5. Marque a opção "Habilitado".
6. No Servidor A, na opção Host/IP Remoto: "IP da RED do servidor B" e vice-versa.
7. Na opção subrede remota: Rede 192.168.X.0/255.255.255.0 do Servidor B.
8. Use uma chave compartilhada "rnpesr" e clique em "Salvar".
9. No menu, clique em "VPNs" e selecione "OpenVPN".
10. Na opção Sub-Rede OpenVPN digite a subrede remota: 192.168.X.0/255.255.255.0 e clique em "Salvar".
11. No menu, clique em "Firewall", selecione "Firewall Settings" e marque as opções "Verde", "IPsec-Red", "OpenVPN-RW", "Show interface colors in rule overview" e clique em "Salvar".

Ao término destas configurações efetivas nos dois servidores, as subredes de ambos deverão estar se comunicando.

Teste do Tunnel IPSec

Para testar a conectividade, abra o "Prompt de Comando" (DOS) na máquina Windows Estação A e execute:

```
C:\ ping 192.168.X0.1
```

Atividade 10.4 – Configurando regras de firewall

Diferente do iptables em texto puro através do shell, no IPCop todas as regras de firewall são configuradas em uma interface web muito intuitiva. Por padrão, o firewall está configurado para permitir acesso de qualquer máquina interna para a rede externa e negar qualquer tentativa de acesso de uma máquina externa para a rede interna.

O objetivo desta atividade é demonstrar o mecanismo de criação e gerência de regras de acesso. Para isso, configure as seguintes regras:

- Permitir o acesso ao serviço SSH do firewall apenas para a máquina 192.168.X.2.
- Permitir que a máquina externa 192.168.X0.2 acesse o serviço OpenVPN (porta 5003) na máquina local 192.168.X.2.
- Logar pacotes ICMP saindo da rede local com destino à rede externa que passem pelo firewall.

Bibliografia

- TERPSTRA, John H.; LOVE, Paul; RECKS, Ronald P. (et alii). *Segurança para Linux*. Editora: Campus, 2004.
- DHANJANI, Nitesh. *Hack Notes: Segurança no Linux e Unix*. Rio de Janeiro: Editora Campus, 2004.
- NETO, Urubatan. *Dominando Linux Firewall Iptables*. Rio de Janeiro: Editora Ciência Moderna, 2004.
- HUNT, Craig. *Linux: Servidores de Rede*. Rio de Janeiro: Editora Ciência Moderna, 2004.
- JAMIL, George Leal; GOUVÊA, Bernardo Andrade. *Linux para profissionais: do básico à conexão em redes*. Editora: Axcel Books, 2006.
- SALMEN, Fadir. *IPSec*. Monografia apresentada na Universidade Estadual de Londrina (2002).
- ROTOLE, Erick Dantas. *Arquitetura IP Security*. Trabalho de curso apresentado na Universidade de Brasília. Disponível em: http://www.cic.unb.br/~pedro/trabs/ipsec.pdf
- FILHO, João Eriberto Mota. *Descobrindo o Linux*. 2ª ed. Rio de Janeiro: Editora Novatec, 2007.
- SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sérgio. *Redes de computadores: das LANs, MANs e WANs às redes ATM*. 2ª ed. Rio de Janeiro: Editora Campus, 1995.
- TANENBAUM, A. *Redes de Computadores*. 3ª ed. Rio de Janeiro: Editora Elsevier, 1997.
- TORRES, Gabriel. *Redes de computadores: curso completo*. 1ª ed. Rio de Janeiro: Editora Axcel Books, 2001.
- SOUSA, Lindeberg Barros de. *Redes de computadores: dados, voz e imagem*. 6ª ed. São Paulo: Editora Érica, 2005.
- RIBEIRO, Uira. *Certificação Linux*. Editora: Axcel Books, 2004.
- NEMETH, Evi; HEIN, Trent R.; SNYDER, Garth. *Manual Completo do Linux: Guia do Administrador*. Editora: Prentice Hall, 2007.

- SIEVER, Ellen; WEBER, Aaron; FIGGINS, Stephen (et alii). *Linux: o Guia Essencial*. Editora: Bookman, 2006.
- RFC 854 – Telnet Protocol Specification
- RFC 951 – The Bootstrap Protocol bootp
- RFC 1631 – The IP Network Address Translator (NAT)
- RFC 1661 – The Point-to-Point Protocol (PPP)
- RFC 1939 – Post Office Protocol (Version 3)
- RFC 2401 – Security Architecture for the Internet Protocol IPSec
- RFC 2402 – IP Authentication Header
- RFC 2406 – IP Encapsulating Security Payload
- RFC 2407 – The Internet IP Security Domain of Interpretation for ISAKMP
- RFC 2408 – Internet Security Association and Key Management Protocol
- RFC 2409 – The Internet Key Exchange (IKE)
- RFC 2616 – Hypertext Transfer Protocol (HTTP/1.1)
- RFC 2131 – Dynamic Host Configuration Protocol
- RFC 2341 – Cisco Layer Two Forwarding Protocol (L2F)
- RFC 2637 – Point-to-Point Tunneling Protocol
- RFC 2661 – Layer Two Tunneling Protocol (L2TP)
- RFC 4101 – Writing Protocol Models
- Guia Gentoo Linux de Segurança. Disponível em: www.gentoo.org/doc/pt_br/security/
- Artigo de Antonio Cláudio Sales Pinheiro. Disponível em: http://www.linuxit.com.br/modules.php?name=Sections&op=viewarticle&artid=450
- Manual de Segurança do Debian. Disponível em: http://www.debian.org/doc/manuals/securing-debian-howto/index.en.html
- Linux Security. Disponível em: http://www.linuxsecurity.com/
- Linux firewall and security site. Disponível em: http://www.linux-firewall-tools.com/linux
- Security Debian Manual. Disponível em: http://www.debian.org/doc/manuals/securing-debian-howto/index.pt-br.html#contents

- Funcionamento do IPSec. Disponível em: http://www.ciscopress.com/articles/article.asp?p=24833&seqNum=6
- Edgard Jamhour. *IPSec: IP Seguro.* Disponível em: http://www.ppgia.pucpr.br/~jamhour/Download/pub/Outros/IPsec.ppt
  http://www.debian.org
  http://www.tldp.org/LDP/nag/node34.html#SECTION004600000
  http://open.nit.ca/wiki/?WvDial
  http://nmap.org/
  http://www.wireshark.org/
  http://www.netfilter.org/
  http://www.tcpdump.org/
  http://focalinux.cipsga.org.br/guia/avancado/ch-fw-iptables.htm
  http://www.cisco.com/warp/public/537/6.html
  http://www.cisco.com/univercd/cc/td/doc/cisintwk/idg4/nd2012.htm
  http://standards.ieee.org/getieee802/download/802.1Q-1998.pdf
  http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/55168.htm
  http://www.abusar.org/vpn/vpn2.htm
  http://mia.ece.uic.edu/~papers/volans/pptpd1.html

Pedro R. Torres Júnior é professor da Universidade Federal do Paraná e coordenador técnico do ponto de presença da RNP no estado do Paraná. Também atua como coordenador técnico do ponto de troca de tráfego (IXP) da região e da Rede Metropolitana de Curitiba. Possui mais de 14 anos de experiência em sistema operacional Linux e mais de 12 anos de experiência com roteamento IP.

Christian Lyra Gomes é formado em Ciências da Computação e tem mestrado em Redes de Computadores pela Universidade Federal do Paraná. Teve seu primeiro contato com Linux em 1997 e em 2000 trabalhou na Conectiva Linux. Em 2001, foi contratado pelo Ponto de Presença da RNP no Paraná atuando na área de gerência de redes e sistemas. Em 2006 assumiu a coordenação administrativa do PoP-PR, cargo que exerce até hoje.

Francisco Marcelo M. Lima é certificado Project Management Professional (PMP) e Modulo Certified Security Officer (MCSO), Mestre em Engenharia Elétrica pela Universidade de Brasília (2009), Mestre em Liderança pela Universidade de Santo Amaro (2007) e pós-graduado em Segurança de Redes de Computadores pela Universidade Católica de Brasília (2003). Atualmente exerce as funções de Coordenador dos Cursos de Redes de Computadores e Segurança da Informação do IESB, e de Analista em TI do MPOG cedido para a Controladoria-Geral da União/PR. Possui mais de 15 anos de experiência na área de Ciência da Computação, com ênfase em Segurança da Informação, Redes e Construção de Software.

Luiz Fernando Ramos Costa é Pós-graduando em Engenharia de Software – UNISUL/SC, formado em Redes de Computadores pela Estácio de Sá, certificado LPI Linux, Novell CLA, DCTS, analista de TI do IFSC e analisa de suporte na POWERSolutions. Mais de 13 anos de experiência em administração de redes, desenvolvimento de softwares e como orientador de cursos para a certificação Linux Professional Institute, atuando principalmente na gerência de redes em grandes corporações e com programação de sistemas web.

esr.rnp.br

LIVRO DE APOIO AO CURSO

O curso fornece o conhecimento prático e teórico necessário para a configuração e manutenção de redes de computadores em ambientes seguros. Apresenta as principais atividades em administração de redes e sugere políticas de segurança. Serão estudados os fundamentos da arquitetura TCP/IP, sua pilha de protocolos e serviços oferecidos. As atividades práticas incluem desde a configuração e monitoramento da rede até a implementação de firewall, NAT, roteamento e tunelamento, utilizando ferramentas como Wireshark, Iptables, IPSec, OpenVPN e rkhunter.

Este livro inclui os roteiros das atividades práticas e o conteúdo dos slides apresentados em sala de aula, apoiando profissionais na disseminação deste conhecimento em suas organizações ou localidades de origem.

ISBN 978-85-63630-21-6